BIM SYNAPSE:
A FRAMEWORK FOR BIM INTEROPERABILITY IN THE CLOUD

A Dissertation
Presented to
The Academic Faculty

by

Kereshmeh Afsari

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Design

Georgia Institute of Technology
August 2017

BIM SYNAPSE:

A FRAMEWORK FOR BIM INTEROPERABILITY IN THE CLOUD

Approved by:

Professor Chuck Eastman, Advisor
School of Architecture
Georgia Institute of Technology

Dr. Dennis Shelden
School of Architecture
Georgia Institute of Technology

Professor Daniel Castro-Lacoutor
School of Building Construction
Georgia Institute of Technology

Dr. John R. Haymaker
Director of Research
Perkins + Will

Professor Shamkant Navathe
School of Computer Science
Georgia Institute of Technology

Date Approved: June 21, 2017

To My Wonderful Father and My Loving Mother

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Professor Chuck Eastman for all the help and guidance that he has given me over the past five years. He is an impeccable researcher and has been a great inspiration to me. I consider myself extremely lucky to have been part of his research team. I especially would like to extend my sincere gratitude to Professor Daniel Castro-Lacouture who supported me on this journey and for his helpful recommendations. He has set an example of excellence as a researcher, mentor, and role model. I would also like to thank Professor Shamkant Navathe for his valuable discussion, feedback, and constructive remarks on my work. I would like to express my gratitude to other members of my doctoral committee, Dr. Dennis Shelden, and Dr. John Haymaker for their time and guidance through this process.

I am truly grateful for the love and constant encouragement of my parents and for the support they have given me in all my pursuits. My heartfelt thanks and appreciations to my loving parents, my twin brother and his wife, and my sister.

I would like to thank the faculty and staff in the College of Design especially my academic advisor, Ms. Robin Tucker, for their support through this journey. I am grateful to my lovely friend, Khatereh Hadi, who has always been an incredible support. Finally, I want to thank all my friends and colleagues in the College of Design especially Dr. Paula Gomez, Fereshteh Shahmiri, Dr. Laura Florez, Dr. Kia Mostaan, Dr. Marcelo Bernal, Dr. Francisco Valdes, Dr. John Fard, Dr. Yongcheol Lee, Dr. Yeliz Kahya, Chen Feng, Dr. Mahsa Nicknam, Dr. Sabri Gokmen, and Matthew Swarts for their support and friendship through the years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AEC	Architecture, Engineering, and Construction
API	Application Programing Interface
BIM	Building Information Modeling
EMPC.1	Precast Concrete Exchange Model number 1
IFC	Industry Foundation Classes
IoT	Internet of Things
JSON	JavaScript Object Notation
MVD	Model View Definition
NBIMS	National BIM Standard
NIBS	National Institute of Building Sciences
PCI	Precast Concrete Institute
REST	Representational State Transfer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WoT	Web of Things
$ref	JSON keyword for references

SUMMARY

In the Architecture, Engineering, and Construction (AEC) industry, collaboration within Building Information Modeling (BIM) process is mainly based on transferring files. BIM data is being exchanged in either vendor specific file formats or neutral format using Industry Foundation Classes (IFC) as open BIM standard. However, since the web enables cloud-based BIM services, it provides an opportunity to exchange non-file based data via the web and over the networks. Alternative BIM data sharing solutions have been developed based on the federation of BIM models with BIM server technologies or using an interchange hub for data exchange in real-time. These solutions face several challenges, are vendor locked, and integrate two or multiple applications to a third new system which is tightly coupled. In addition to scalability issues, these data sharing technologies make the collaborating applications dependent upon each other which end up with high complexity. In fact, current cloud-based interoperability solutions do not provide a loosely coupled system with the flexibility to reduce dependencies among collaborating applications. Therefore, the main objective of this research is to propose an interoperability framework that supports a network-based BIM data exchange for loosely coupled collaboration in the cloud.

This research emphasizes that there is a need to reshape BIM collaboration in the cloud by using web technologies. This study indicates that Cloud-based Building Information Modeling needs to deploy major components of the cloud interoperability including the APIs, data transfer protocols, data formats, and standardization to redefine BIM dataflow in Cloud-BIM applications. BIM Synapse framework proposed in this

research utilizes web technologies- as the enabler for a cloud-based collaborative process- to restructure current BIM dataflow. BIM Synapse deploys cloud interoperability features and IFC data model to address current challenges of BIM data exchange in the cloud and provides a loosely-coupled network-based data interoperability solution for Cloud-BIM. The study also applies the proposed framework on BIM collaboration in the conceptual design process of precast concrete buildings and evaluates the correctness, accuracy, completeness, and consistency of the BIM Synapse framework. BIM Synapse framework has a major contribution to standardization of Cloud-based BIM data exchange and can enable the integration of the Internet of Things (IoT) - that requires network connectivity and provision of resources through the Web of Things (WoT)- with the BIM process. The study also recommends required revisions to the IFC specification so that the IFC schema can perform as the basis for Cloud-BIM interoperability.

CHAPTER 1. INTRODUCTION

This chapter gives an overview of the study and briefly explains the opportunities and challenges that the combination of Cloud computing and Building Information Modeling (BIM) has introduced to the architecture, engineering, and construction (AEC) industry. The significance of developing an interoperability framework for Cloud-based BIM is also discussed in this chapter. Moreover, the study of the role of standards for model-based data exchange within BIM-based collaborations is the focus of this chapter, and it highlights existing challenges of BIM data exchange. This chapter also outlines the gap in knowledge, research objectives, and its hypotheses as the basis for the investigation.

1.1 Cloud Computing and BIM

Cloud computing as defined by the U.S. National Institute of Standards and Technology (NIST), "is a model for enabling ubiquitous, convenient, on-demand access to a shared pool of configurable computing resources" [1]. Cloud computing relocates the computing process and data from desktop to large data centers [2]. Thus, Cloud computing aims at making a better use of distributed resources for multi-user collaborative interaction and combining them to achieve higher performance so that large scale computation can be managed [2, 3, 4].

As a rapidly emerging technology, Cloud-based BIM has become a new research area in the AEC industry since 2010 [5, 6]. It is believed that Cloud-BIM could provide project partners and design disciplines with better capabilities to share and exchange design data requirements and solutions [7]. Cloud computing can perform as an effective means

to overcome current BIM challenges by providing real-time access to a pool of data, on-demand access to computing resources (e.g. storage, servers) and applications, higher performance and potentially better interoperability [8]. Besides, Cloud-based BIM technology is known as a cost-effective alternative to the current state of data exchange and storage [8]. Since potential values of Cloud-based applications such as efficiency and low-cost have been recognized, the combination of BIM and Cloud computing is believed to be a promising trend [6, 8]. Cloud-BIM is anticipated to change the AEC industry although the technology is still relatively new [5].

The development of Cloud-based BIM services has already created a new direction in BIM implementation to support collaborative BIM data generation and consumption among project partners [9]. Therefore, Cloud-BIM technology is believed to provide higher levels of information interaction and further cross-disciplinary collaboration [5, 4].

1.2 Cloud-BIM Development and Challenges

Because of the potential advantages of Cloud computing, BIM applications are steadily moving to the Cloud and BIM web services and Cloud-based apps are gradually gaining increased popularity [9, 10, 11]. Some examples of the Cloud-based BIM solutions are GRAPHISOFT BIM Explorer (BIMx), Autodesk BIM360, BIMServer.org developed by TNO – Netherlands Organization for Applied Scientific Research- and the University of Eindhoven, ONUMA System [12, 9, 5] and Trimble Connect. These Cloud-based solutions are being developed mostly in isolation and by different vendors without considering how they should eventually interoperate across platforms. Thus, there is a need to reconsider the approach to interoperability of new Cloud-BIM services otherwise they

will suffer from the same issues as interoperability challenges in conventional desktop-based applications [13].

Interoperability is the key to the success of Cloud services implementation [5]. However, the challenge is that making multiple Cloud-BIM applications interoperate would be very difficult when they are developed by different vendors running on different Cloud platforms [14]. Besides, existing methods of BIM data storing and transferring based on neutral files or a centralized database as will be discussed in chapter 2, have not been successful in addressing fast data retrieval and maintaining data consistency [6]. Therefore, the issue of data exchange and interoperability for Cloud-based BIM applications requires further studies [9].

Current Cloud-BIM interoperability approaches can be categorized to three groups in terms of dataflow architecture. First, manual file transfer that is currently a common way of exchanging BIM data across applications. Project data can be exported and shared in the form of vendor specific formats or neutral format using IFC standard [6]. Second, BIM server technologies in which server-based BIM solutions known as model collaboration systems [11] have provided a central BIM service with a single-sourced data server accessible for project partners [6, 12]. These model server technologies utilize information directly from the models and are intended to improve multidisciplinary collaboration [11]. The issue with BIM server technologies is that their implementations are still limited [11] and hence have faced scalability issues [9]. Third and more recent category includes Data Interchange Hub such as Flux [15] that can automate dataflow between certain applications. This type of solution currently has a very limited implementation and supports very few design applications to be connected via the hub. Moreover, in this approach the

inter-connection of applications relies on the hub solution and its capabilities although supported software packages can exchange data directly.

Standardization is an effective solution to address the interoperability issue of the Cloud services [16, 17]. Cloud-BIM applications are being developed in isolation and the standardization of BIM in the Cloud as discussed in the following chapters has suffered from the limitations of data serialization and the lack of established Cloud-specific standards.

Most importantly, existing Cloud-BIM interoperability solutions face and mix different challenges [18] and have not fully utilized the potential of Cloud computing by implementing a standardized network-based data transmission process as will be discussed in this study. Particularly, the architecture of Cloud-BIM interoperability has not addressed a loosely coupled collaboration. The loosely coupled collaboration which will be discussed in more details is a Cloud architecture that is based on autonomy in access policies and resource management [19]. Therefore, it is critical to rethink the interoperability of Cloud-BIM applications.

1.3 BIM Data Interoperability

In the AEC industry, project parties such as architects, engineers, construction team and fabricators most often work on several platforms to generate separate BIM models. Therefore, BIM data needs to be transferred between software applications during collaborations in design and construction process. Information sharing is the starting point of collaboration and it requires applications to be able to exchange data regardless of vendors and data formats. To achieve this, in the AEC industry, building data is described

in Industry Foundation Classes (IFC) specification as an ISO standard to support a neutral data format and to facilitate cross-platform BIM interoperability [20, 21]. IFC data model describes building data and provides a means to define building components and processes in a publicly available data schema as an open standard [20] to address cross-platform BIM data exchange. IFC data schema is represented as an EXPRESS schema specification [22]. EXPRESS is an information model specification language based on ISO standard and specified as part of the STEP standard for product model data exchange [23] and uses the STEP physical file structure. IFC data model is alternatively represented in XSD schema specification using the XML document structure (i.e. ifcXML). The specification of ifcXML ensures to handle the same data as represented in EXPRESS specification of IFC data model [21]. The ifcXML file structure with "ifcXML" or “ifx” or “.xml” extension, is the XML document structure and can be used as a data format in Web-based systems.

1.3.1 Model View Definition (MVD)

National Building Information Model Standard (NBIMS) specifies a set of interoperable standards to support building data exchange for a project’s lifecycle. For Building Information Modeling, the NBIMS goal is to support transparent and standard information exchange [24]. Guidelines are defined by the NBIMS to develop information exchange standards for all phases of the project including design, construction and operation based on all project parties including architecture, engineering, construction, fabrication, and facilities management [25]. The specification process for data exchange has three major steps as shown in Figure 1. First, a group of domain experts as the workgroup develops a functional specification i.e. Information Delivery Manual (IDM); Second the IDM is translated to a technical specification for software vendors to implement

which is based on IFC schema; Third the exchange specification is implemented, tested and certified. In fact, the exchange specification forms a set of Model View Definitions (MVDs) to define requirements for standard data exchange for tasks during design, engineering, and construction [24, 26]. Upon completing these steps, this exchange specification can be deployed by users in the industry [26, 24, 27].

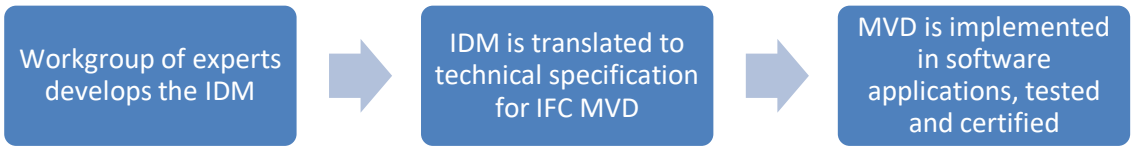


Figure 1 MVD specification process

MVD provides specification of the data that needs to be exchanged. The MVD specification is prepared within a format that can be used and implemented by software companies. Particularly the specification for an IFC MVD describes detailed mapping to IFC data objects to be used by software implementers [24, 26]. Within the MVD, each Exchange Model is the detailed functional specification for a specific BIM data exchange [25] performed at a specific stage of the project with a specific purpose of model-based collaboration. Domain experts should review the MVD specification both from the sender and receiver point of view to ensure the MVD contains all the information needed for the BIM exchange [20].

1.3.2 Current BIM Dataflow

The goal of an MVD is to map exchange requirements for one or multiple model exchanges to a data schema, like the IFC schema [28]. An IFC MVD documentation provides information needed for the exchange of BIM model referencing the IFC standard

[29]. An IFC view definition or an IFC MVD specifies a subset of the IFC schema that is needed to satisfy one or many exchange requirements of the AEC industry [28].

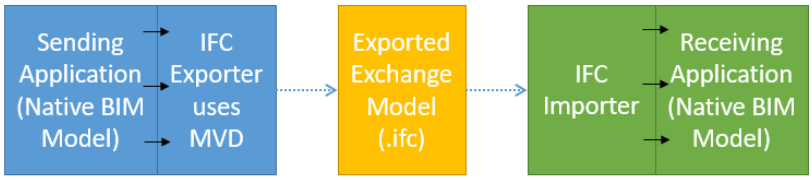


Figure 2 Current process of BIM data exchange using IFC MVD

MVD consists of one or multiple exchange requirements which must be provided by the sender of BIM data to support work in the receiving application. If the IFC model is generated using a specific MVD, it means the IFC model contains the data which is required for that specific exchange purpose e.g., for conceptual model coordination, structural analysis, clash detection, etc. The NBIMS [29] also explains that the sender and receiver of BIM data need to agree on exactly which information to exchange and the receiver can run a quick test based on the MVD requirements to verify whether the sender of BIM data has sent enough information. In other words, first, there should be a joint agreement by sender and receiver of BIM model for the contents of an MVD. Then, the exporter would generate the model using MVD requirements and the exported model must be tested and validated to ensure it contains required data. Similarly, when the BIM model is imported in the receiving application, it should be tested and validated against the required MVD. However, in practice typical workflow in using MVD is as follows: Sender generates the BIM model in the sending application and wants to export the BIM model for specific use in receiving application (e.g. for structural design purposes). Sending application, as illustrated in Figure 2, uses an IFC MVD exporter to export the BIM model

to an IFC file. Then, the sender passes the IFC file to the receiver of data. Upon receiving the exported IFC model in an IFC file format, the receiver of data imports the model in the receiving application which is usually a new environment. The receiving application uses an IFC importer module to translate the IFC file to native binding. This way the interoperability between sending and receiving applications with different software architecture and data structures can be managed.

When the MVD specification is developed, it should be implemented by at least two software applications i.e. sender and receiver applications to make it usable for the end users [28]. However, a major challenge in this BIM exchange process is the validation of imported model [18]. In fact, after the MVD implementation, it is required to complete third party testing and certification of these software applications to ensure a reliable data exchange for the end users. The challenge is, software certification testing for import is very complex because it relies on the evaluation of how the exchanged model is being interpreted and used in the importing application, i.e., receiving application [28]. In this dataflow, the focus has been on the export side than the import side as the final consumer of BIM data.

In addition, in this process the request for data happens outside of the BIM model and the receiver of BIM data needs to wait until the sender of data exports the BIM model as a file and passes it to the receiving party. This means that the receiver party or application does not deal with getting the data directly until the file is exported, sent and becomes ready for importing in the receiving application.

1.4 Motivations and Problem Definition

Cloud-BIM is known as the second generation of BIM development and studies suggest that it will produce a major change across the industry although the technology is still relatively new [5]. Cloud-BIM has created a new direction in BIM implementation and development to support BIM data generation and consumption among members of the project [9]. By applying Cloud Computing in BIM services, Building Information Modeling can achieve a higher performance with a relatively low cost [6]. Therefore, Cloud-BIM technology is believed to lead to higher levels of information interaction and can provide an effective cross-disciplinary collaboration [5, 4]. Besides, interoperability is the key to the success of Cloud implementation [5] and data exchange is critical for successful implementation of Building Information Modeling [4]. However, making multiple Cloud-based applications interoperate has been a major challenge [14].

Collaboration in the architecture, engineering, and construction (AEC) industry mainly relies on file transfer while BIM data is being stored and exchanged in the form of files with several formats [6, 11]. The formats of these files could be either vendor specific or neutral format using IFC [6] as an open BIM standard. IFC as an ISO standard and an open BIM standard has facilitated cross-platform BIM interoperability. However, file-based BIM data exchange has created a disconnected process that leads to inconsistencies in design and construction. Current Cloud-based data exchange technologies- which will be discussed later in details- such as solutions used in BIM server technologies or data interchange hub have not fully utilized the potential of web technologies to support a network-based interoperability solution.

On the other hand, since the web enables Cloud-based services, its role is more than performing just as a web-based user interface for Cloud-based solutions. Most importantly,

the web can facilitate data sharing and interoperability [13]. In fact, Cloud offers an opportunity that software packages can be connected via the web, therefore these services can manage information communication in a different way [14] than the conventional file-based system. Particularly, data sharing among multiple Cloud applications is based on the type of the collaborative environment [19] and the architecture for a loosely coupled collaboration reduces dependencies between Cloud applications and simplifies the data exchange. In a loosely coupled collaboration, components of any Cloud applications can change without affecting other collaborating applications and without affecting the interoperability.

Thus, the main motivations for this study are: first, growing development of Cloud-BIM and benefits of the Cloud-based Building Information Modeling; second, the significance of Cloud interoperability, importance of data exchange in Building Information Modeling as well as the interoperability challenges; third, the benefits and drawbacks of current BIM interoperability facilitated by IFC specification but with a file-based approach; and, forth, the potentials of the web technologies for enabling a network-based interoperability solution within a loosely-coupled collaborative environment. This study highlights that currently there is a gap in research regarding the identification of technologies that can assist with Cloud-BIM interoperability solutions. This research seeks to answer the following questions:

1. What are the current limitations of existing Cloud-based interoperability for BIM collaborations in Cloud?
2. What approaches and techniques of web technologies can assist with BIM data transmission to address a network-based data exchange in the Cloud?

3. What dataflow architecture should be applied for BIM data exchange in the Cloud to address current challenges?
4. How can IFC specification - as the open BIM standard in the AEC industry- enable data interoperability in Cloud-BIM applications?
5. How can MVD- as the information exchange standard- facilitate BIM data exchange in the Cloud?

Therefore, this research underlines recent developments and opportunities to address fundamental challenges of BIM data exchange. This study first outlines the features and issues of current Cloud-BIM solutions, especially regarding data exchange approaches. Then, the study will point out advances in data transmission in Cloud as well as Cloud interoperability components that can be deployed to address current Cloud-BIM data exchange challenges. This research also investigates different types of collaboration architecture for Cloud-based data sharing solutions to recommend the most effective architecture for data sharing in Cloud-BIM. It eventually specifies a data exchange architecture as an underlying basis for Cloud-BIM interoperability and describes the components of this architectural model with its evaluation and implementation specifics explained in details in the following chapters.

1.5 Hypothesis and Research Objective

The main objective of this research is to develop and evaluate an interoperability framework for exchanging model-based data in Cloud-BIM applications that use potentials of web technologies to enable a network-based BIM data transmission, based on the following three hypotheses:

1. IFC specification as BIM open standard can be used to guide a network-based BIM data interoperability in the Cloud.
2. BIM data interoperability can be achieved with a loosely coupled system in the Cloud that reduces dependencies between sender and receiver of model-based BIM data.
3. The receiver of the BIM model can use MVD specification to request and receive BIM data directly.

This research uses Cloud standards as well as IFC open standard to address dataflow issues for model-based BIM exchange among Cloud applications. The study aims at achieving a non-file-based data exchange of BIM models over a network with loosely coupled integration in Cloud.

1.6 Overview of the Proposed Framework

To address the study hypotheses, a framework for Cloud-BIM data interoperability is proposed in this study as BIM Synapse. The study specifies the components of BIM Synapse framework. It uses IFC data in web compatible format, specifies the design for a loosely coupled solution with data request capability on the receiver adhering to IFC specification, and undertakes multiple data validation strategies to evaluate the proposed framework.

BIM Synapse is the proposed framework for interoperability of Cloud-BIM applications. The significance of developing this framework as well as the architectural model of BIM Synapse is discussed in the following chapters. The reason behind the naming of BIM Synapse is based on the notion of synapse in human nervous system.

Synapse is a structure that enables the communications among neuron or nerve cells with a gap junction that passes signals [30]. It is where the communication among neurons happens as illustrated in Figure 4 and for that reason, it has similarity in function with the proposed framework.

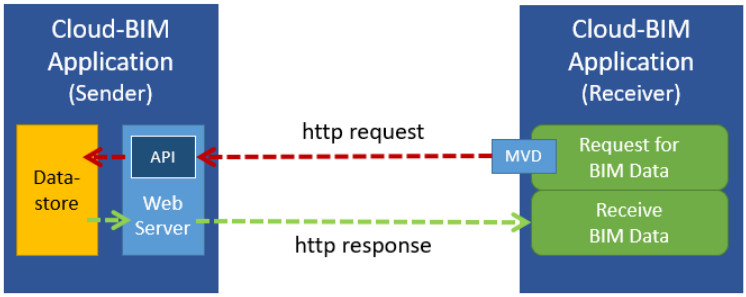


Figure 3 Overview of BIM Synapse framework

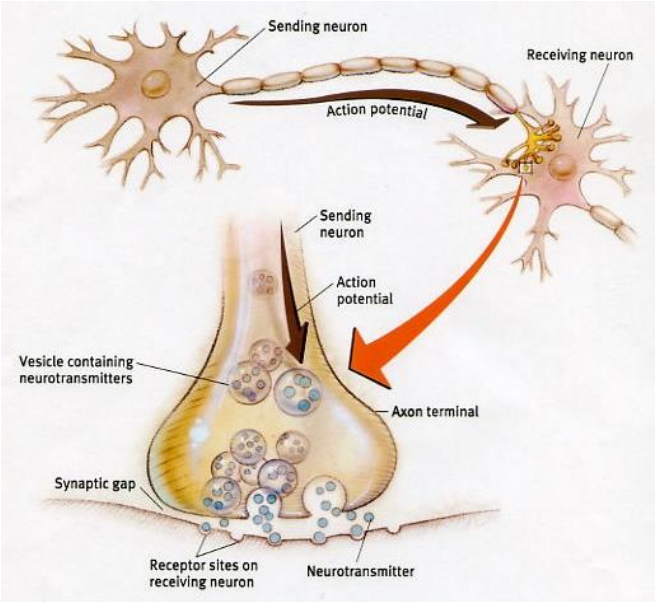


Figure 4 Neural pathways and communication through connections i.e. synapse- Source: Rowland Hall [31]

1.7 Framework Evaluation Criteria

An empirical evaluation has been undertaken to demonstrate the feasibility of the proposed framework. The study has implemented the framework, developed test cases and then applied several validation techniques on both the data from the test cases and components of the framework to evaluate the framework. Chapter 6 explains the evaluation method and metrics for evaluating the framework through the application of test cases for the study experiment, framework implementation, data validation and evaluation.

Four evaluation criteria have been established for different aspects of requirements specification that is explained in chapter 6. The criteria as shown in Figure 5 can be translated to the following questions.

- Correctness- is the data translation correct, is the data being used valid, and is the data being exchanged represent the expected BIM model?
- Accuracy- does the proposed framework perform at the required level of accuracy and do the components of the framework operate accurately?
- Completeness- is the BIM model received over the web protocols using the proposed framework complete?
- Consistency- does the proposed framework address the research problem and is it consistent with the research requirements?

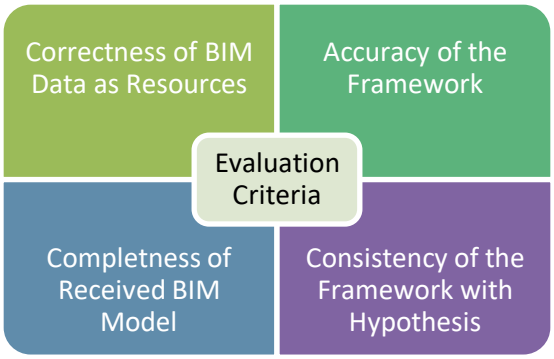


Figure 5 Four evaluation criteria in the research

1.8 Scope of Research

This research focuses on model-based data exchange in Cloud-BIM applications so transmission of other types of data other than BIM models is not in the scope of this study. Moreover, the research emphasizes on standardization of BIM data exchange in the Cloud and thus, the use of industry accepted standards is critical. The implementation of the framework focuses on using the framework to send correct BIM data with proposed methodology and to receive a complete BIM model in the form of web compatible data formats on the client side. Thus, the integration of data within the client application and guidelines for mapping BIM data to native binding and vice versa is not in the scope of this research.

The research introduces a new architectural model for exchanging BIM models using web technologies and although the framework can benefit other aspects of BIM process such as model validation, it is not the focus of the study and similarly, rule checking is not included in the scope of this study. Also, the proposed framework is applied to the

precast concrete domain and an example of precast concrete BIM exchange model is discussed so other domains and exchange requirements are not in the scope of this study. Similarly, data security and ownership for Cloud-BIM technologies are open research areas but are not included in this research.

1.9 Structure of the Thesis

This thesis describes an investigation to interoperability of BIM data in the Cloud and introduces a new data flow and architectural model to improve BIM data transmission within a collaborative process. The research is designed in eight following chapters.

Chapter 1 gives an overview of the thesis and introduces the study. It formulates the research questions by articulating the problem. This chapter also explains the study hypotheses as well as the research objective. It discusses a summary of the proposed framework and briefly explains the study evaluation criteria.

Chapter 2 explains how BIM data transmission is managed in current Cloud-BIM applications and what specific challenges exist in current systems. It highlights the gap in research and the need for an effective network-based BIM data exchange. This chapter studies the existing methodologies for Cloud-BIM data integration and indicates the advantages and disadvantages of current Cloud-based BIM interoperability approaches.

Chapter 3 discussed the opportunities in web technologies and explains a new interoperability architecture. It introduces BIM Synapse as the framework for BIM interoperability. The chapter explains the dataflow in BIM Synapse that supports a network-based BIM data transmission and can address domain-specific exchange

requirements by deploying IFC specification. In addition to standardization as an effective means to enable Cloud-BIM interoperability as well as the use of the HTTP protocol to manage communication between clients and servers, data serialization and a RESTful API are considered as major features of the BIM Synapse discussed in chapter 4 and 5 respectively.

Chapter 4 highlights the importance of choosing the proper data serialization and discussed the need for the JSON implementation of IFC specification that is based on both JSON schema and IFC schema. ifcJSON4 schema developed here is a valid JSON schema that can guide the creation of valid ifcJSON documents to be used for web-based data transfer within BIM Synapse framework.

Chapter 5 focuses on Cloud APIs and introduces an architectural style for the design of a standardized API for Cloud-BIM that is scalable and loosely-coupled and can support standardized BIM data exchange in the Cloud based on web technologies.

Chapter 6 presents the results of a rigorous empirical analysis of the BIM Synapse and provides a proof of concept in the form of application, implementation, validation, and evaluation of the proposed framework. Focusing on comparing theory to reality within an empirical evaluation, this chapter explains the evaluation metrics, design, and execution of the experiment, domain application, technical implementation, designing test cases and data validation. It also explains the evaluation of the research hypothesis.

Chapter 7 explains the impact of the research and discusses the impact of proposed data representation, improved collaborative process, and standardization of BIM interoperability achieved in this research. The chapter also explains the recommended revision to IFC specification to improve the IFC schema for its usability in Cloud-BIM

interoperability. Also, this chapter explains the study contributions to both body of knowledge and the state of practice.

Chapter 8 explains the challenges and limitations of this research. The chapter also summarizes the results of this research and revisits the research questions to point out how the research has answered the research questions.

In addition, Appendix A shows the ifcJSON4 schema developed and used in the study to validate the ifcJSON documents as BIM data.

CHAPTER 2. BACKGROUND

This chapter investigates how building data transmission is managed in current Cloud-BIM applications and what specific challenges exist in current systems. Here, the study represents the existing methodologies for Cloud-BIM data integration and features of each technique are specified. The chapter indicates the strengths and weaknesses of current Cloud-based BIM interoperability architectures in terms of data transmission requirements for Cloud-based BIM applications and provides a comparison of existing systems. In addition, the study points out the challenges of cross-platform BIM data transfer in making multiple Cloud-BIM applications interoperate. The challenge in current data transmission strategies highlights the need for an effective network-based BIM data exchange to address a collaborative BIM workflow in the Cloud.

2.1 Cloud-BIM Data Integration

Available Cloud-BIM data integration solutions on the market mainly address the idea of centralization of BIM data. The emergence of server-based BIM solutions has provided a central BIM service to all project members [6, 11]. These model server technologies establish a single-sourced data server [6] that uses data directly from the models (i.e. sub-models) and with a consolidated model it can improve multidisciplinary collaboration [11]. BIM server implementations are still limited [11] but BIM server technology has changed BIM work-sharing with a database driven approach [9, 11, 12].

Current BIM server technologies on the market provide functionalities like querying BIM models as well as graphical interfaces for sharing and viewing BIM models in a team on a centralized platform. Examples of these BIM server technologies are

GRAPHISOFT® BIMcloud® for design process, Trimble Connect for project collaboration, Autodesk A360 for design delivery and BIM360 for construction project delivery. Also, Autodesk Revit Server is the server application for Revit Architecture, Revit Structure, and Revit MEP and performs as a server-based work-sharing platform for Revit projects. In addition to proprietary BIM server technologies, BIMServer.org as an open source technology has been developed by TNO and the University of Eindhoven [12]. BIMServer.org architecture is shown in Figure 6. BIMserver centralizes the information of a project with a core that is based on IFC and shares its information to client applications through some interfaces. This solution is not a file server but its architecture interprets IFC data from a file and stores it in a database. Therefore, it can merge and query the model and it eventually generates IFC files [32]. Other examples of IFC-based BIM server implementation are IFC model server developed by VTT Building and Transport and SECOM Co., Ltd. and also, EDM Model Server which is an IFC model server based on EDM developed by Jotne EPM Technology [6].

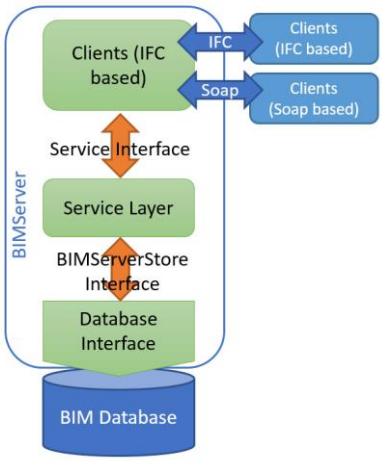


Figure 6 BIMServer.org architecture

These Cloud-based BIM servers could be theoretically a good solution however in practice they have faced major challenges [6] such as scalability and robustness [9]. BIM server technology overcomes the issue with full-size model synchronization [9] while the integration of BIM models in the Cloud provides a network-based data exchange [5]. However, the technology is not robust [9]. In this type of Cloud-based collaboration some data management issues exist especially in combining models [5] and in sub-model extraction [6]. Also, these solutions require more powerful Web-based operating systems, file-sharing platforms and hardware controllers [5]. Studies identified that existing BIM servers require functional and performance improvements [6, 11] as well as coordination with vendor specific format data [6].

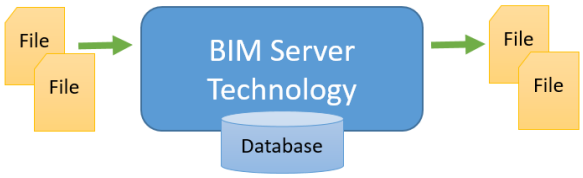


Figure 7 Architecture of a BIM server solution

Most importantly, the input and output of these systems, as shown in Figure 7, is heavily based on files. For instance, input and output of BIMServer.org is based on IFC files. Also, the existing BIM servers cannot work with the decentralized, heterogeneous and dynamic design data; thus, they cannot support the whole lifecycle of the project [6].

In addition to BIM server solutions, Flux project which was initially started in late-2010 at Google[x], Google's research lab, is a Cloud-based collaboration tool for the design

process to assist architects, engineers, and contractors with exchanging data [18]. Unlike conventional file-based data transfer, Flux acts as an interchange point for sharing project data such as design, analysis, and schedules. Flux plugins should be installed on design software applications to automate data transfer to and from Flux. Currently, Flux works with a limited number of applications such as Rhino/Grasshopper, Excel, Revit/Dynamo, and SketchUp. Figure 8 illustrates the dataflow between applications and Flux.

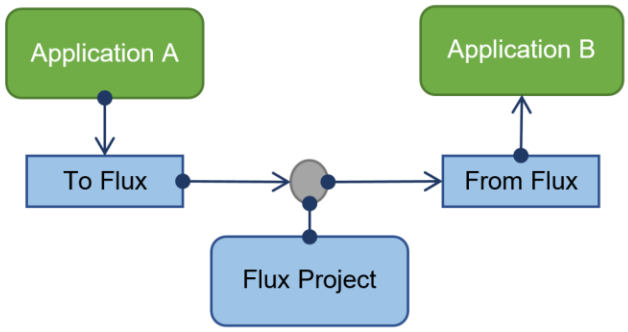


Figure 8 Flux dataflow for design data exchange

2.2 Comparison of BIM Data Integration Strategies

The study of the existing Cloud-BIM data integration methodologies suggests that there are three main categories of BIM data integration in the Cloud that allows cross-platform data exchange. Comparison of these three methodologies is summarized in Table 1. Advantages and disadvantages of each method are specified.

Table 1 Comparison of Cloud-BIM data integration methodologies

Cloud-BIM Data Integration Technique	Pros	Cons
Manual File Transfer	Could be based on established standards and MVDs	Includes only one-way data transfer and repeats for each design iteration
	Can use neutral data format	Export and import validation issues
BIM Server Technologies	Centralizes BIM data	Suffers from scalability and performance issues, and data format issue
	Improves collaboration in an integrated Model	Depends on the server platform and not completely connected to the origin of data
Data Interchange Hub	Automates dataflow	Supports very few applications
	Reflects design changes in real-time within each BIM application	Depends on the interchange platform

2.2.1 Manual File Transfer

Manual file transfer can be based on BIM open standard as well as vendor specific formats. Using neutral file format based on IFC data model and established MVDs can be potentially advantageous to help AEC industry stick to a common language to improve collaboration. However, file-based data transfer is a one-way communication that should be repeated for each design iteration to include constant design changes. In addition, in this process, there are methods to validate the exported model against the initial exchange requirements but how to validate the imported model is still vague. This dataflow is shown in Figure 9. Existing file-based data transfer technologies are not suitable for BIM applications because of incapability of managing data redundancy and inconsistencies [6].

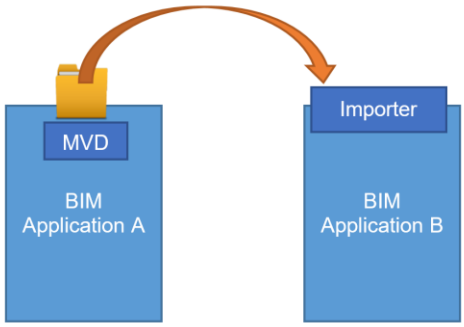


Figure 9 Dataflow for manual file transfer

2.2.2 BIM Server Technology

BIM server centralizes BIM data in a database and can improve collaboration in an integrated model. But the technology is limited and has performance and scalability issues in dealing with complex and big projects. The integration dataflow for this methodology is shown in Figure 10 and Figure 11. BIM server technology provides a centralized and accessible repository for the project and can provide access to project data almost anytime and anywhere throughout the building lifecycle. However, since these solutions only support a limited number of data formats as inputs and outputs (i.e. mainly data formats supported by one vendor), the centralized model can be disconnected from the origin of data if the data is provided by a different vendor whose format is not supported. In this case, data transfer should be tackled with exporting and importing files (Figure 10). This makes the integration to rely on manual file transfer. In BIMServer.org for instance, a user (e.g. an architect) should check in a file-based IFC instance model and the model will communicate with server in the Graphical User Interface (GUI). Similarly, the outputs of the service are based on files too if needed to be transferred after analysis, simulation, etc.

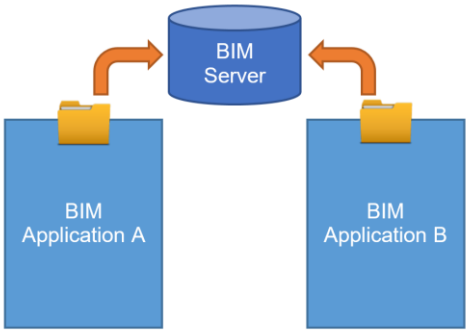


Figure 10 Integration dataflow for BIM server technology in data transmission between applications developed by different vendors.

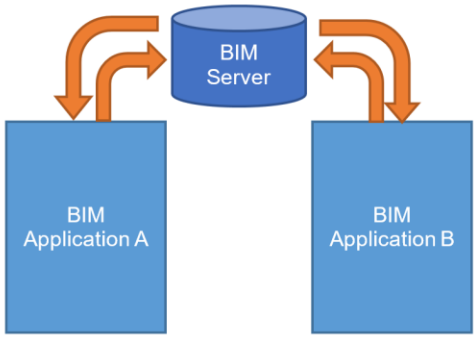


Figure 11 Integration dataflow for BIM server technology in data transmission between applications developed by one vendor.

2.2.3 Data Interchange Hub

DIH technology such as Flux project can automate dataflow between certain applications. This solution can reflect the changes in real-time while each user and application controls when to synchronize data with the project. Data Interchange solution allows users to work in isolation and share their changes when they are ready [15]. But this solution currently supports very few design applications to exchange data. In addition, the applications are dependent on Flux although the model can be updated on each platform on its own. The integration dataflow for Data Interchange Hub solution is shown in Figure 9.

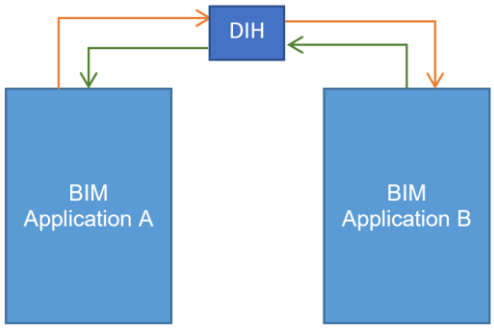


Figure 12 Integration dataflow for Data Interchange Hub (i.e. Flux) solution

In addition to these BIM data integration solutions that are developed to enable cross-platform data exchange and to provide cross-disciplinary collaboration solutions, there are other third party solutions developed for a specific purpose. These cloud-based third party solutions extract data from the BIM models in BIM authoring tools through some plug-ins to publish the model in a new environment so that they can provide additional specialized services such as analysis.

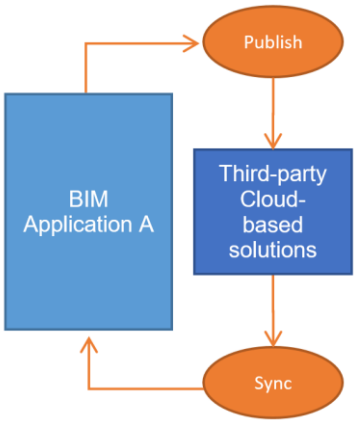


Figure 13 Dataflow for third-party Cloud-based BIM solutions

As distributed services, these Cloud-based solutions can eventually synchronize the modifications applied in the model back to the original BIM authoring tool. For instance,

BIM Assure provides BIM model checking service for validating the model accuracy and completeness. Also, Assemble Insight provides cost estimation solutions and analyses such as value engineering. These solutions are currently limited in exchanging and mapping of data structures in different formats and can only deal with BIM data from very few BIM authoring tools and data formats, mainly supporting one vendor specific data structure. Since this type of Cloud-based BIM solutions is not considered as consolidation products and are not supposed to provide integration platforms, cross-platform data transfer solutions or solutions to connect applications, they are excluded from the comparison in this research. The overall dataflow for these Cloud-based solutions is illustrated in Figure 13.

2.3 Related Efforts and Research

BIM Service interface exchange (BIMSie) is the effort for standardization of Application Programming Interfaces (APIs) for BIM web services [10]. BIMSie highlights that web-based BIM services each develop their own API different from other application APIs and therefore, many custom interfaces are required to connect BIM applications [33]. BIMSie aims at standardizing API calls to address API standardization to allow online BIM services to connect with each other. After the connection is made, the data can be exchanged in files such as in IFC, COBie, or similar [10, 33]. Some examples of methods (i.e. API calls) that are standardized in BIMSie are calls like 'addProject', 'download', 'getAllRevisionsOfProject', 'newRevision', 'getAccessMethod', 'login', 'abortTransaction', 'commitTransaction', 'getDataObjectsByType', 'getProgress', etc. When all online applications use the same API calls it allows to create an automated interaction among these services. The BIMSie has four modules: core or the 'ServiceInterface',

'NotificicationInterface', 'RemoteServiceInterface', and 'Authinterface'. In addition, there is the 'LowLevelInterface' module for more advanced operations such as changing an IFC model. BIMSie specifically works with BIMserver.org [10]. Although BIMSie can address the standardization of BIM APIs to let BIM web services APIs connect and interact, its model-based data exchange strategy follows the traditional file-based data transfer while in design iterations the BIM file will be automatically uploaded when the model changes in BIMServer.org.

Juan and Zheng [4] proposed a Cloud-based Open BIM framework for building information interaction. They illustrated the architecture of cloud deployment pattern and the information interaction process. Their framework includes Infrastructure layer that maps to Infrastructure as a Service (IaaS), data layer and model layer that map to Platform as a Service (PaaS), service layer and application layer that map to Software as a Service (SaaS). The infrastructure layer mainly contains all kinds of physical resources and virtual resources. Their data layer contains the project information in the whole life cycle of buildings. The information can be divided into three categories, namely the building information model based on the IFC, unstructured data and information which is stored in a distributed network environment. Their model layer mainly includes the specifics of the business process. In their service layer, they propose using XML encoding for data transmission, where a service request is sent by the user and the server information feedback will be received accordingly with no need to download the complete IFC data files. Their application layer deals with the human-computer interaction and includes the interaction among project participants and the execution of projects [4]. Their framework is very general and deals with organized Cloud-BIM deployment. It does not provide a

specific interoperability solution in the service or application layer although it suggests using XML-based IFC data as a web compatible data format.

Zhang et al. [6] propose a framework of Cloud BIM service that virtually integrates distributed servers through a Cloud computing platform. They highlight the inefficiencies of existing file-based data transfer in BIM process and emphasize that although BIM server technologies within a centralized service have addressed some of the problems, these services have faced data permission problems within project participants. They have proposed to use multiple enterprise servers instead of a centralized server and build an integration platform with distributed processing capabilities of clusters of servers to link these enterprise servers. As the integration platform, they developed a BIM integration and service platform (BIMISP) based on IFC and Cloud computing. This platform deals with data exchange and sharing and includes three main modules: distributed data storage using SQL servers with IFC data parsing, distributed data integration, and sub-model extraction that is built on Hadoop HBase. HBase is column-oriented and allows storing and querying structured IFC data. They arranged the database tables based on independent IFC entities with GlobalIDs and stored resource entities as attributes in the form of binary data in HBase's cells. Their comparative efficiency analysis between a centralized BIM server using a relational database and their proposed cloud BIM server with HBase database shows efficiency advantages in their proposed service platform [6]. Their BIMISP platform adds a third platform to the cycle of BIM applications to provide data integration capabilities and makes enterprise servers and BIM applications dependent on BIMISP.

Curry at al. [13] proposed the use of linked data for Cloud-based building data services to create an integrated and connected information graph for managing a building.

They argue that linked data formats based on W3C standards allow building data to be combined with data from other domains and stakeholders. They explain that IFC data by itself cannot enable interoperability with systems outside of the AEC domain. They propose the use of Resource Description Framework (RDF) standard as a model for data linking, sharing and reusing on the web. They highlight the need to reflect on the design of Cloud-based services in terms of their interoperability capabilities. They have developed a case study of a linked building data to demonstrate how building data and other related data can be integrated using an example of energy management in the building [13] within their Building Energy Explorer [34]. They have specified a new vocabulary as RDF entities and manually mapped to IFC entities from an IFC file using ifcOWL, a Web Ontology Language (OWL) for IFC specification. To merge data from different data sources such as energy sensors and BIM model, they identify similar resources that appear in multiple sources and specify the relationships between equivalent resources. They also anticipate that the future work will focus on interoperability of RESTful services [13]. The use of semantic web technologies can improve interoperability, knowledge representation, and semantic data sharing [35, 36]; however, proposed framework [13] which is based on converting IFC files to capture the building data creates a disconnected environment with data inconsistency issues. In addition, the RFD vocabulary that is aimed at providing a shared understanding of data in the schema level is not standardized.

2.4 Cloud-BIM Interoperability Challenges

The study so far indicates that Cloud-BIM data transmission for cross-platform collaboration faces several challenges including standardization, data interdependency, and data access and security.

- Standardization: There is a lack of Cloud specific standards for BIM interoperability. With the growing number of Cloud-BIM services developed by several providers, standardization among these service providers become important [37]. The Open Cloud Manifesto, an important effort in Cloud standardization, emphasizes on open standards for Cloud computing [38]. Therefore, industry-wide open standards like IFC specification should be expanded to address the requirements of Cloud-based applications.
- Data interdependency: Current BIM integration solutions that deal with Cloud interoperability are either addressing model federation in a centralized platform or interconnecting a limited number of design applications on premise through a new Cloud-based solution with the help of plug-ins. These solutions integrate two or more systems to a third new system, instead of creating a loosely coupled aggregation, shown in Figure 14, where each system remains self-contained.
- Data access and security: Collaborative nature of Cloud-BIM data integration causes security challenges such as liability and BIM model ownership [5, 8]. User authentication and authorization is key to successful deployment of Cloud-BIM and therefore Cloud identity management and role-based user access becomes significant which require advancement in trust and privacy preserving techniques [8]. Providing a safe service for Cloud-BIM applications and integration solutions is critical. The issues of dealing with the data security, ownership and stability for Cloud-BIM technologies are still open research areas.

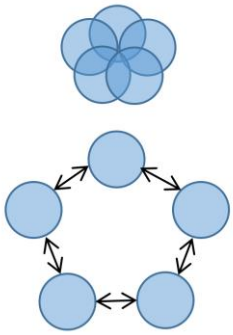


Figure 14 Tight coupling (top) vs. loose coupling (bottom) between Cloud applications

Existing architectures for Cloud-BIM data integration face several challenges in model-based data exchange and have not fully exploited the potential of the Cloud towards a loosely coupled integration. Therefore, there is a lack of an architecture that redefines the dataflow utilizing web-based technologies as major enablers of the Cloud. In addition, while vendor specific data formats are quite diverse, these are based on multiple and different data schemas. At the same time, data standards for Cloud-based cross-platform data exchange purposes are limited. IFC data model which describes building data provides a means to define building components and processes in a publicly available data schema. As an industry-wide open standard, IFC schema definition has not become the basis for Cloud-BIM integration solutions although it can ensure a common understanding of building data across the applications and disciplines. Most importantly, there is a lack of effort in revisiting the NBIMS process to consider the requirements of Cloud-based data transmission and data integration workflow with Cloud-based use cases.

CHAPTER 3. ARCHITECTURAL MODEL FOR BIM SYNAPSE

This Chapter explains the nature of data exchange and data interoperability in Cloud and studies the architectures for different types of Cloud collaborations. It outlines the opportunities that exist in Cloud computing technologies as well as BIM data standardization in the AEC industry to address BIM interoperability. This chapter also discusses the framework for BIM Synapse as an architectural model for Cloud-BIM interoperability and describes the components and features of BIM Synapse framework.

3.1 Data Transmission in the Cloud

Cloud technology is an internet-based computing. Cloud services provide architectural principles and software specifications to connect computers using standardized internet protocols [16]. The internet is populated by a lot of computers that are either clients which are terminals to access resources, or servers that provide data illustrated in Figure 15. Different operating systems can exchange data over the internet because they all use a set of rules and a common standard known as Transmission Control Protocol/Internet Protocol (TCP/IP). For data transmission between client and server computers connected to the internet, a set of rules or protocols are used that is defined by TCP/IP [39].

Cloud computing is based on TCP/IP and in fact, Cloud could not be achieved without standard interconnected protocols. TCP/IP provides Cloud computing with reliable delivery of data between remote applications with a connection-based service [40]. To exchange data, TCP/IP subdivides data in packets before it transmits the data thus, the information should be split and packaged to be sent and received [39].

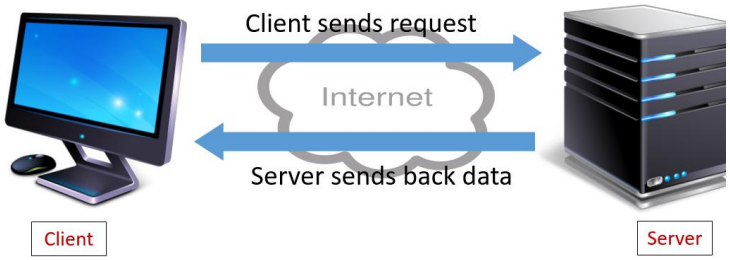


Figure 15 The Client-Server model

3.2 Cloud Interoperability Features

There are several categories of Cloud services such as infrastructure, platform, and application. Based on the services that the Cloud solution provides, there are three major types of Cloud computing models as IaaS or Infrastructure as a Service, PaaS or Platform as a Service, and SaaS or Software as a Service [41, 2, 3, 40].

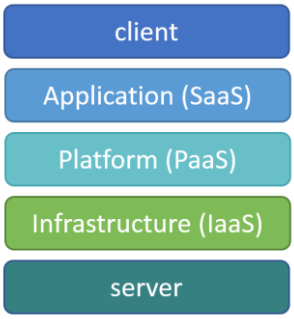


Figure 16 Layers of Cloud Architecture

Figure 16 illustrates the distinct layers of Cloud architecture: a client includes computer software and hardware for application delivery, a Cloud application delivers SaaS, platform services or PaaS provide a computing platform using the Cloud infrastructure provided by IaaS, and a server contains computer software and hardware for the delivery of Cloud services [2].

In Cloud computing the term “interoperability” might sometimes refer to “portability” which is the ability to move a system from one cloud platform to another [41, 3]. Interoperability which is discussed in this research, is considered as the ability of data exchange and integration. Here, interoperability deals with what is known as “enabling products/software components to work with or integrate with each other seamlessly to achieve a desired result” [42]. The problem is, each Cloud provider incorporates Cloud computing with a self-contained set of conventions, data formats, and application programming Interfaces (APIs) and to allow Cloud services interoperate identification of major Cloud interoperability components is critical [43]. There are four key features of Cloud interoperability including the API, data format, data transfer protocol, and standards.

3.2.1 Cloud APIs

Capabilities of Cloud services are represented through interfaces which can be accessed through the Application Programming Interfaces (APIs). Therefore, if in Cloud applications a common set of APIs with agreed terminologies are used, these applications can interoperate [44]. “Cloud APIs specify how software applications interact with a Cloud-based platform where these applications can be deployed” [17]. These APIs define how applications can request information from the platforms and how to use their facilities. Cloud APIs can be in the form of web services e.g. based on Representational State Transfer (REST) or Simple Object Access Protocol (SOAP), application dependent protocols, high level programming languages, or remote calls (e.g. Java RMI, AMF) [17, 38, 16, 41]. However, Cloud services that are developed separately by different vendors have different APIs and this makes the interoperability difficult [16]. Such differences require human intervention to connect and interact with Cloud services.

3.2.2 Data Transfer Protocols

As mentioned, Cloud computing is based on TCP/IP and standard protocols have made Cloud computing possible [40]. Common TCP/IP protocols are HTTP (Hyper Text Transfer Protocol) as a protocol designed to allow the transfer of Hypertext Markup Language (HTML) documents, FTP (File Transfer Protocol) for high-speed disk-to-disk file transfers, and SMTP (Simple Mail Transfer Protocol) as an internet mailing system [45]. FTP deals with the transmission of files between computers but HTTP is based on request-response activity. World Wide Web traffic mostly uses the HTTP protocol because HTTP uses the most bandwidth across the Internet [45]. As illustrated in Figure 17, HTTP takes care of the communication between a web server and a web browser. It first establishes a secure connection between a server and a browser. Then, a client that is running an application on the web browser can send requests to the connected server and upon request, the server sends data that resides on a repository or a data-store to the client [40].

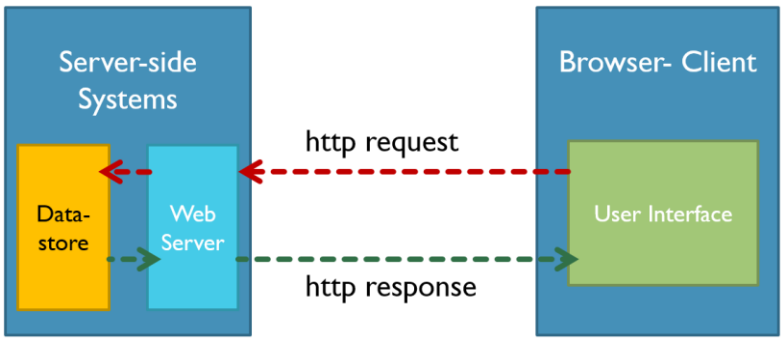


Figure 17 HTTP request/response model

3.2.3 Data Formats

Access through the Cloud APIs is supported by several data formats. Choosing the proper data serialization is critical due to the data exchange increase over the internet. The data format is even more significant in mobile devices because these devices use limited resources and are bandwidth limited [46]. Two most common text-based approaches of data serialization are eXtensible Markup Language (XML) and JavaScript Object Notation (JSON) that have been used widely in web applications for data transmission [47, 46]. XML and JSON have different features. XML has a prescriptive grammar and stores all data in the closed tag with the data indexed by labels. JSON on the other hand is a lightweight data exchange format and uses a text format independent of the language and has higher parsing efficiency than XML [47]. More recently, binary data serialization formats are being used such as Google's Protocol Buffers (ProtoBuf) and Thrift developed by Facebook [46] as well as Apache Avro developed as a Hadoop subproject [48]. These are extremely lightweight, and fast to serialize and deserialize [46, 48]. The studies show that the size of binary-based serialized data is better than XML or JSON-based serialization [48].

3.2.4 Standards

Standardization is an effective solution to address the interoperability issue of the Cloud services [16, 17]. There are many Cloud standardization projects such as Open Cloud Computing Interface (OCCI) and Open Cloud Manifesto [38, 17]. Some of these efforts explain standardizing parts of a Cloud computing solution such as data access and others deal with standardizing how parts of a solution should work together [41]. Although

existing standards can support Cloud interoperability, there are different levels of system interoperability that should be carefully considered such as technical interoperability (i.e. deals with exchanging data), semantic interoperability (i.e. deals with exchanging meaningful data), and organizational interoperability (i.e. deals with participating in multi-organizational business processes). Existing standards can address technical interoperability but might not be able to guarantee semantic or organizational interoperability [41]. To address semantic and organizational interoperability, domain knowledge based on established schema and use cases are required. In this regard, the need for a standardized API is the main semantic interoperability requirement since a standardized Cloud API created and supported by the Cloud vendors can resolve compatibility conflicts between Cloud services and can allow one Cloud system cooperate with another Cloud system [44, 16, 41]. Therefore, the main step of managing interactions of Cloud resources and services is establishing a standardized API. In addition, a common Cloud data model should be used as the basis to describe Cloud systems components such as resources, services and APIs [44, 38].

3.3 Cloud Collaborations

Collaboration and data sharing among multiple Cloud applications is based on the type of the collaborative environment [19]. There are three types of collaborations:

a) Federated collaboration supports a long-term interoperation and needs the collaborating Cloud applications to have mutual dependence and trust [19, 49]. This architecture sets a global policy framework consistent with local policies of each collaborating Cloud [19].

b) Loosely coupled collaboration is a Cloud architecture that is based on autonomy in access policies and resource management [19]. Interactions among collaborating Cloud applications are guided by local policies [19, 49]. Resources in each Cloud can be created virtually and authorized for autonomous sharing among collaborating Clouds.

c) Ad hoc collaboration deals with the service needs at the time of deployment [49]. Since the service requirements might not be available at the beginning, access to the resources might be denied. Therefore, to support dynamic interoperation, specific implementation of authentication and authorization mechanism is required and secured collaboration needs to be developed on a user basis [19].

To evaluate these types of collaboration, four major metrics can be used: 1) degree of interoperation which is the level of service and resource sharing among collaborating Cloud applications, 2) autonomy which is the ability of each Cloud application to perform its local operations with no interference from collaborating Cloud applications, 3) degree of privacy which quantifies as to what extent a Cloud application should disclose its local policies, constraints and rules. 4) verification complexity which specifies how complex the verification of the correctness of all the constraints is in the integration [19, 49]. In the work of Almutairi et al. [19] the evaluation metrics for federated, loosely coupled, and ad hoc collaboration are analyzed and assessed. Figure 18 diagrams the tradeoffs based on the analysis in [19]. The diagram shows that although federated collaboration can provide the highest degree of interoperation among these three systems but because of generating interdependencies, it causes low degree of privacy and autonomy for the collaborating

applications. Also, while ad hoc collaboration can support higher degree of privacy, it has limited provides low degree of interoperability compare to two other systems.

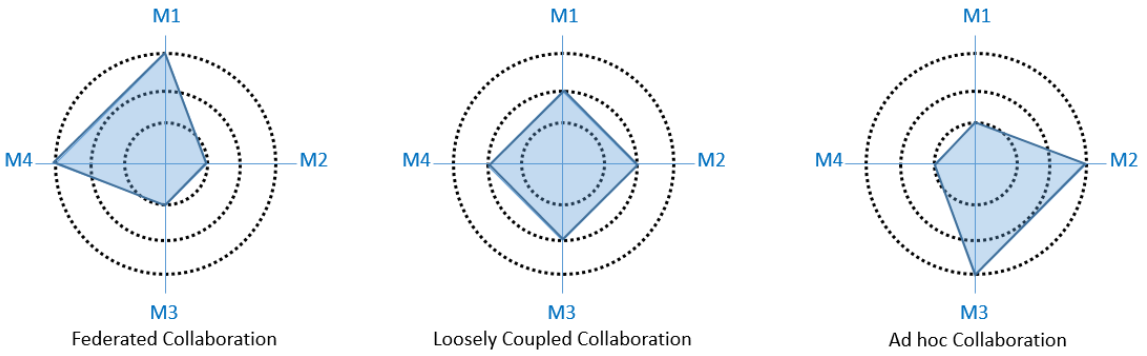


Figure 18 Tradeoffs for Cloud collaboration types considering evaluation metrics analyzed in [19] (M1= degree of interoperation, M2= autonomy, M3= degree of privacy, M4= verification complexity)

Loosely coupled Cloud collaboration are consistent with federated collaborations and ad hoc collaboration. Federated Cloud collaboration though can end up with high complexity because of its need for generating a global policy framework out of many heterogeneous and sometimes conflicting policies thus, it is not scalable [19]. A federation can also be loosely, partially or tightly coupled [49]. In ad hoc collaboration although a high level of privacy is supported, it does not federate credentials and has the least level of interoperation among all [19].

Overall, the architecture for loosely coupled collaboration reduces dependencies between Cloud applications as well as components of the applications. So, data exchange can be significantly simplified. Also, loosely coupled collaboration provides a reasonable degree of autonomy and privacy with reasonable verification complexity as shown in Figure 18. In a loosely coupled collaboration, components of any Cloud applications can change without affecting other collaborating applications thus, resource sharing will be

intact since collaborating Clouds follow the same resource management. Therefore, this study suggests the architecture for loosely coupled collaboration to support Cloud-BIM data interoperability.

3.4 Cloud Integration Solutions and Their Limitations

As explained in the previous sections, Cloud-BIM data interoperability has faced several challenges and has not fully utilized the potential of Cloud computing by implementing a standardized network-based data transmission process. Therefore, this study has also investigated alternative methodologies used in Cloud interoperability solutions developed for other domains. Data interoperability among Cloud applications has been an ongoing challenge [50]. There are several Cloud integration solutions that provide complete integration services such as providing interfaces in collaborating applications (i.e. both Cloud applications and on premise) as well as the design of the integration to support custom integration in Cloud [51]. Examples of Cloud integration solutions include WebSphere Cast Iron Cloud Integration (IBM Cast Iron), Dell Boomi by AtomSphere, and MuleSoft.

WebSphere Cast Iron provides mapping among different data structures stored on different database systems. It allows integration to be executed on its Cloud service or an on premise solutions and devices [50]. Its integration approach follows a direct and pair-wise federation with data synchronization and it does not support data sharing within a loosely coupled collaboration [14]. On top of its platform, it deploys and configures plug-ins. The plug-ins for ad hoc collaborations can also be developed by Integration specialists.

Dell Boomi provides consolidation services to utilize full capabilities of Cloud. It connects Cloud and on premise applications without the need for any appliances, software or coding [51]. It also supports data exchange and mapping of different data structures that are in different formats such as XML and EDI [50].

MuleSoft has an integration package that provides integration solutions for Cloud and on premise applications for developers to connect applications to exchange data. To manage the API, it supports many data transfer protocols as well as data delivery style [51]. It can integrate applications that are based on different technologies and allows developers to create customized Cloud connectors.

Overall, current Cloud integration solutions allow the integration of two or multiple application to a third new system and they do not allow for a loosely-coupled collaboration [50]. These solutions are tightly coupled and if any of the application changes any of its data format or interface requirements, this might cause interaction failure and the system needs to be updated. They all follow a configuration approach rather than programmatic interoperability [50]. Thus, these systems only identify the options available for data sharing while programming approach deals with a comprehensive understanding of collaborating applications and data transfer protocols for data sharing. A few similar solutions are underway for the AEC industry to address Cloud-BIM integration such as Autodesk Quantum. It is also predicted that this type of Cloud integration will be largely developed in near future for the AEC industry.

These Cloud interoperability solutions suffer from vendor lock-in and several problems with resource management and data sharing while each one uses their own

interoperability framework and data schema [17]. Data integration and interoperability have traditionally seen several challenges of inconsistencies. While Cloud computing is believed to resolve limitations of conventional services, it has not been very effective in current data integration solutions. Data integration must expect data to conform to a common schema [52] to address interoperability challenges.

3.5 BIM Synapse Architecture

Current challenges highlight the significance of a new framework for Cloud-BIM interoperability. Moreover, identification of major components of Cloud interoperability provides main aspects that should be considered in implementing Cloud-based BIM applications so that they can exploit the potentials of the Cloud to address BIM interoperability in cross-platform network-based data transmission.

In addition, as discussed current file-based BIM data exchange process shown in Figure 19 is based on using the MVD in the sender application (i.e. application A) to generate an exchange model which is a file mainly in ".ifc" format. This file is then passed to the receiving application (i.e. application B) which uses a translation module as the IFC importer to interpret and visualize the imported model in a native environment.

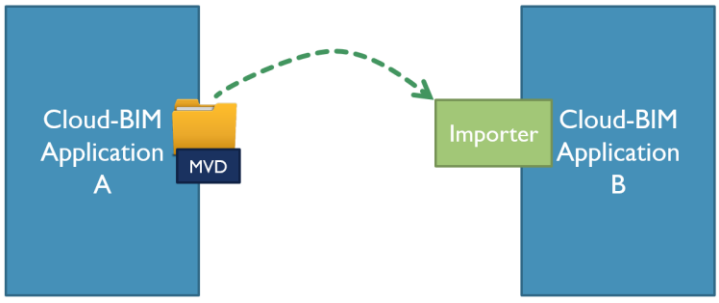


Figure 19 Manual file-based BIM data exchange process

Using an MVD to generate the exchange model makes the collaboration smooth by addressing the expectations regarding data requirements. Also, the use of an open standard data schema following an IFC MVD ensures data interoperability in cross-platform collaborations. However, in this process, the request for data happens outside of the BIM model, usually through emails, agreements for project check-ins, or other methods of correspondence. For instance, the project collaborator who needs the BIM model and is the receiver of data (e.g. structural design team) requests for a specific BIM data (e.g. architectural model) based on a specific exchange in an MVD, e.g., a conceptual design from the sender of the data (e.g. architectural design team). In this process, the receiver of BIM data needs to wait until the sender of data exports the BIM model as a file and passes it to the receiving party, meaning that the receiver party or application does not deal with getting the data directly until the file is exported, sent and becomes ready for importing in receiving application. To address the challenges of conventional file-based data transfer, an alternative approach should be utilized. This new approach should incorporate four major features of Cloud interoperability identified earlier for a loosely coupled collaboration. This study proposes an architecture for the implementation of an API based on standards and proper profile of data format and protocols.

However, Cloud APIs provided by Cloud-BIM vendors have not been standardized yet. BIMSie project [10] has worked towards introducing a standardized Service Interface (i.e. API) for Cloud-BIM solutions. This standard Service Interface is supposed to automate interaction between Cloud-BIM applications by introducing API for Cloud-BIM applications with standardized methods [10]. In the BIMSie initiative when the APIs are standardized and Cloud-BIM services are connected, the data exchange is file-based using

industry standards as IFC files which mean the data exchange is not network-based and simply follows traditional file-based exchanges. The BIMSie project is an important step towards facilitating Cloud-BIM interoperability by standardizing and connecting Cloud APIs. But the issue of file-based data transfer has not been addressed yet and full potential of Cloud technology is yet to be deployed.

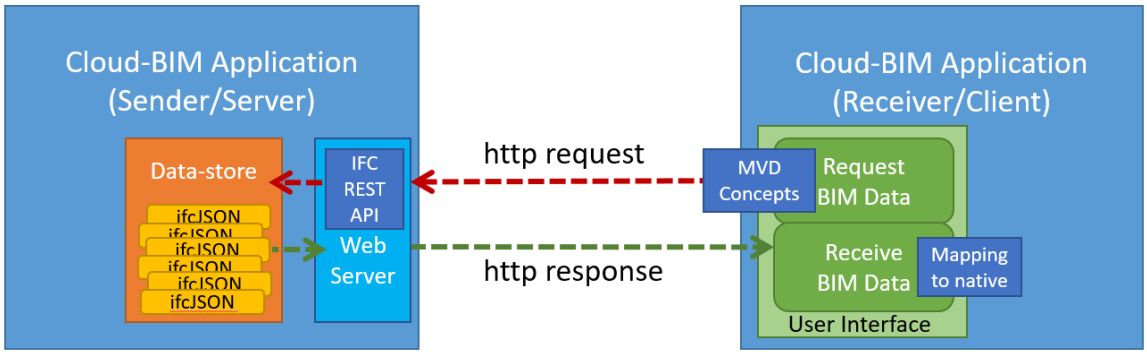


Figure 20 BIM Synapse Architecture

Therefore, to take advantage of the opportunities exist in web technologies as well as with Cloud interoperability for the benefit of Cloud-BIM data exchange, an alternative architecture should be utilized. This new architecture is called BIM Synapse and an outline of the BIM Synapse architecture is shown in Figure 20. In this architecture, data for Cloud collaboration is stored in a data-store within Cloud application "A" which is the sending application (e.g. architectural design application) and can be accessed through an API with a standard set of methods and terminologies. The data can be interpreted by Cloud application "B" which is the receiving application e.g. structural design application. The API in fact, provides an on-demand access to the pool of data on application "A". Accordingly, data request happens only in application "B" that becomes the client for application "A" and uses a data exchange requirements (i.e. MVD) to request for the BIM

data. The data exchange requirements in this process follow the rules defined in the MVD specification.

3.6 BIM Synapse Components

Major components of BIM Synapse architecture are based on features of Cloud interoperability as follows.

- **The API**: BIM Synapse architecture suggests that each application API in Cloud service provider should implement a set of standardized resources stored in a data-store with an appropriate data serialization format that is retrievable over HTTP. This API needs to follow a common data model. This study suggests using IFC specification as industry-wide standard data model to address semantic interoperability. The API design will be discussed in chapter 5.

- **Data Transfer Protocol**: BIM Synapse architecture defines a dataflow for exchanging model-based data between Cloud-BIM applications that is different from the conventional methods such as manual file transfer. This approach uses HTTP calls for cross-platform communication. It is based on a simple request/response mechanism. Thus, in this architecture the receiver of data as shown in Figure 20 directly deals with getting the data instead of waiting for the sending application to export data in the form of files. It should be noted that this research emphasizes on BIM data retrieval and therefore, in BIM Synapse the focus is on GET request. Other HTTP operations like PUT, POST, DELETE deal with changing the data directly in the source or removing data directly from the source which is not the case in BIM data exchange process among collaborating parties;

thus, it is not in the scope of BIM Synapse. In this thesis, the only concern is getting the data directly in the receiver side not modifying the origin of the BIM data. In BIM data exchange process, the collaborating parties cannot change the origin of the data directly; for example, a structural engineer is not allowed to modify the architect's model directly on the architect's source model, instead they communicate their recommendations and required modifications within established data exchange process. The detailed description of GET operation in BIM Synapse is explained in the following chapters.

- **Data Format**: In BIM Synapse architecture, the data should be serialized in an efficient compatible format based on common agreements and terminologies. Since IFC is the common standard for open BIM, IFC specification can provide the capabilities to capture domain knowledge as well as understanding with a common data schema. Currently ifcXML is the only IFC data representation that is compatible with Web technologies. However, limitations of XML-based documents should be considered. Web applications are typically represented in the form of Asynchronous JavaScript and XML (AJAX) and Web services. Unlike Web services, AJAX applications are aimed at enhancing the user experience where data transmission speed is very important [47]. When AJAX was first introduced, XML was used widely until it showed inadequacies because of its data structure with redundant tags, larger size and the low efficiency of its analysis in serializing and deserializing data [47, 48]. Therefore, data serialization approaches such as JSON and binary as discussed earlier, should be considered as alternatives to achieve higher efficiency especially in the applications of AJAX technology. The

next chapter outlines the implementation of ifcJSON, a JSON serialization of IFC data model for web-based data transfer.

- **Data-store for BIM Data:** In BIM Synapse architecture, the data-store performs as a repository to persist BIM data. The data serialized in appropriate format needs to be stored and retrieved to and from the API backend. RESTful services use POST, GET, PUT, and DELETE methods of HTTP to correspond to create, read, update, and delete (i.e. CRUD model in database domain) resources but this is only possible if REST resources are designed in the way that they are required to be within the service [53]. This will be discussed in chapter 5.

- **MVD Integration:** In BIM Synapse, since the receiving application initiates the data request, MVD implementation happens on the receiver side. The client uses MVD specification to request for a set of REST resources that corresponds to the exchange requirements in the MVD definition. Upon receiving the data, the MVD-based resources will be mapped to native bindings. This will be discussed in more details in the following chapters.

This architecture supports real-time data exchange while the collaboration is loosely coupled enabled by a RESTful API design as will be explained in chapter 5. In other words, BIM Synapse will not be tying an application to an integration system like a BIM server technology, a data interchange hub, or a Cloud integration platform. The design for the REST API in BIM Synapse and details of its components is discussed in more details in chapter 5, 6, and 7.

3.7 BIM Synapse Features

In conventional BIM dataflow, if a structural designer who is using an application as a receiving application, needs to request a specific information from another discipline such as architectural design team that is using the sending application, the structural engineer should ask the architect to export a model containing required data from the sending application and pass the file to the engineer and then the engineer can import the file in the receiving application to get access to the required data. By using BIM Synapse architecture and the methodology specified in this study, while all project parties are using Cloud-based BIM applications, the structural engineer who uses receiving application only needs to connect to the API of the sending application that the architect is using. Then, all the engineer needs to do is to directly request for BIM data within the receiving application and the engineer will immediately receive data represented in web compatible formats. This way to retrieve the required data, receiver of the data (e.g. structural engineer) interacts directly with the actual BIM model available in the data-store that resides on the sending application. This approach will reshape BIM collaboration process by introducing a new dataflow. Also, the receiver of the BIM data (e.g. structural engineer) has the responsibility to review and approve that the correct set of inputs have been used. Thus, this dataflow could address liability challenges too.

In BIM Synapse, the architectural model for BIM interoperability as illustrated in Figure 20, the MVD implementation will become an effort in the receiver side (i.e. receiving application) and not in the sender side (i.e. sending application). This means that while data is available on and provided by the sender side (i.e. sending application), data request is managed on the receiver side (i.e. receiving application) thus, it will be directed based on the exchange requirements when the receiver of data requests for it. This approach

provides the receiving application and receiving project parties with more control over BIM data exchange. In addition, by considering the MVD on the receiver side, data validity focus will be in the receiving application (i.e. receiving application).

BIM Synapse uses a network-based BIM data transmission and can address the questions regarding what data to request in the receiving application by deploying IFC MVD specification, how to encode and store BIM data to be transmitted properly by deploying web technologies, and how to utilize the received data by using IFC data model. These features will be explained in the following chapters.

CHAPTER 4. DATA SERIALIZATION

This Chapter highlights the need for JSON implementation of IFC specification and introduces ifcJSON Schema and its data content. The main objective of this chapter is to outline how IFC specification can be represented in JSON format. Therefore, the study explains the implementation of the IFC standard as a JSON schema to guide the creation of JSON documents. The ifcJSON documents can be used for web-based data transfer as an alternative to XML documents. Since current IFC specification release is IFC4 Add2, the implementation of ifcJSON4 schema is specified and guidelines for generating and validating ifcJSON documents are described. Additionally, this chapter describes an implementation of the ifcJSON4 schema in a use case within the precast concrete domain by indicating the data content for a precast building element with its corresponding geometry representation, product placement, and owner history data. The analysis of results indicates that ifcJSON4 schema developed here is a valid JSON schema that can guide the creation of valid ifcJSON documents to be used for web-based data transfer within BIM Synapse framework.

4.1 IFC Schema and Data Serialization

The study so far has highlighted challenges in current model-based data exchange and pointed out that current methodologies have not fully exploited the potential of the Cloud. Most importantly, while vendor-specific data formats are quite diverse, they are based on multiple and different data schemas, and data standards for Cloud-based cross-platform data exchange purposes are limited. In the AEC industry, building data such as objects and processes is described in IFC data model schema to support a neutral data format for BIM tools interoperability. IFC schema defines a set of generic building objects

with associated attributes and properties as well as multiple shape definition methods for the objects [20]. IFC data model provides the basis for a common understanding of the building processes and the required information results from their execution [54]. IFC represents an open specification by introducing object classes and provides a useful structure for data sharing among applications [55] in an AEC project.

IFC specification has a data schema that is represented as an EXPRESS schema specification. ISO 16739:2013 consists of the IFC data schema in an EXPRESS schema specification, and reference data for the description and definitions of property and quantity names [22]. EXPRESS itself is an information model specification language which is defined as ISO10303-11 by the ISO TC184/SC4 committee [56]. It is developed as part of the STEP standard for product model data exchange [57]. The IFC exchange file structure, with the extension ".ifc" or ".stp", is known as "STEP Physical File" (SPF) format. It is an ASCII file format using a clear text encoding of product data for exchanging IFC data between different applications [56, 58, 59]. Current IFC release is IFC4 Add2 [60].

IFC data can be encoded in multiple file formats for various purposes [21]. In fact, data serialization is defined as encoding objects or translating data into a format that can be stored in a file, memory or a database or that can be sent to other application. In fact, a proper data serialization format can affect data transmission rates and performance significantly [61]. So far, there is alternatively an XML Schema specification for IFC data model (i.e. ifcXML) represented as XML definitions [21]. The ifcXML file structure, with ".ifcXML" or ".ifx" or ".xml" extension, is the XML document structure [56]. The XML schema is automatically created from the EXPRESS representation of IFC schema by a language binding described by "XML representation of EXPRESS schemas and data",

defined as ISO10303 part 28 edition 2. Therefore, based on this procedure, ifcXML ensures to handle the same data consistently as IFC-EXPRESS and also the data files can be converted bi-directionally from ".ifc" to ".ifcXML" [56, 62]. In ISO-10303 Part 28 Edition 2 standard the mapping from EXPRESS to XML schema is guided by a configuration file that controls the specifics of the translation process. This configuration file is supposed to be standardized and published for each version of the IFC schema to generate ifcXML [58].

The EXPRESS IFC schema, the SPF format and ifcXML formats are each a particular implementation of the ISO standards [58] i.e. ISO 16739:2013. But so far JSON schema specification for IFC data model has not been standardized.

4.2 The Need for JSON-based IFC Specification

Many studies have shown that JSON has been successful to replace XML as data exchange format in Web services [63, 64]. JSON is easy for computers to parse and generate and its syntax is also human readable [61, 47]. JSON uses a text format that is independent of the language so its format is different from XML format that has closed tags [47]. JSON data format supports high scalability and it creates more compact models than XML [64, 63]. Many research projects in the last few years have processed and analyzed JSON data and most of these studies compared JSON with XML data in performing within web services as summarized below.

- XML-based web service runs with low efficiency [63] although XML language has good specification [47]. Using JSON-style format for data exchange, compared to XML, can improve the performance of web service applications [63].

- Results of a case study indicate that JSON is faster and uses fewer resources than the same XML data [61]. JSON performs better than XML in being parsed, being serialized and being deserialized [63, 47] XML objects are analyzed as Document Object Mode (DOM) which takes a long time [47]. XML requires extra libraries to retrieve data from DOM objects [61]. But JSON objects are analyzed as string arrays which can perform much faster [47].

The importance of performance and resource utilization needs to be considered in choosing between JSON and XML formats [61]. On the other hand, Asynchronous JavaScript and XML (AJAX) is a web technology to transfer data between a browser and a server asynchronously and has several advantages over the classic web applications [65]. Initially, when AJAX was introduced XML language was used widely for AJAX development. But soon it was realized that XML is inadequate when applied to interactive pages and the efficiency of the page will be reduced significantly when using XML data [47]. AJAX reduces response time, server load, and bandwidth of web applications [65]. AJAX applications have good support of JavaScript and since JSON is native to JavaScript, it has been largely applied as an alternative to XML [47]. Because JSON is directly supported inside JavaScript, it can be easily used in JavaScript applications to provide a significantly better performance over XML [61].

While JSON has become an obvious choice over XML for web services, so far there is a lack of studies on implementing IFC specification using JSON serialization. In fact, despite the many advantages of JSON, there has not been any implementation of IFC specification based on JSON data exchange format. Therefore, to use the benefits of JSON over XML in the building industry data exchanges, this study proposes the development of

“ifcJSON”. This study investigates how IFC data can be represented in JSON format so that it can be used as an alternative to ifcXML.

Besides, as BIM data is gradually moving to the Cloud, several efforts in translating vendor specific BIM data to JSON format have already been initiated. For instance, vA3C is an open source, a browser-based 3D model viewer for BIM models in the browser which translates BIM data from Revit, Grasshopper, 3DS Max, and Sketchup to JSON data by using a JSON exporter plugin in each of these applications [66]. This JSON serializer uses proprietary JSON schema definition to generate JSON documents and mainly focuses solely on geometry data. Another example is the Autodesk Forge Viewer that displays 3D and 2D models in a browser. While this viewer translates over 70 file formats (such as AutoCAD, Fusion 360, Revit) into a JSON file format [67], it follows a proprietary JSON schema definition that is different from JSON schema used in the other solutions like in vA3C viewer. This study emphasizes the need for a standardized JSON schema based on an industry-wide open standard i.e. IFC schema and not based on a specific vendor schema convention. So, this study focuses on translating IFC specification to JSON schema and JSON document rather than serializing vendor specific BIM data into a JSON format. This study proposes ifcJSON as the standardized JSON encoding of BIM data based on both JSON schema and IFC specification.

One major effort in serializing IFC specification into JSON format is the work in BIMserver.org [12] which is also applied in some of the experiments in the ELASSTIC project (i.e. Enhanced Large Scale Architecture with Safety and Security Technologies and Special Information Capabilities) as described in its recently published report [68]. BIMserver centralizes BIM models of a project with a core that is based on IFC and stores

it in a database [18]. In BIMserver, a set of Service Interfaces has been defined such as JSON interface for interaction with the BIMserver to facilitate connecting to the BIMserver from a browser. The problem is, BIMServer documentation does not explain its JSON schema implementation methodology and validation approach and does not explain JSON document validation against the defined schema either. The detailed study of the source codes, schema definition and example files in BIMserver [69, 70] shows that although BIMserver uses some sort of converting IFC to JSON data, the schema is not a well-structured IFC-based JSON schema for several reasons. Firstly, this schema is not developed according to the JSON schema specification [71] that is the latest Internet Engineering Task Force (IETF) documentation published for JSON specification. JSON schema specification will be explained in the following sections. A JSON value i.e. value of the “type” keyword in JSON schema must be an object, array, number, or string, or one of the three literal names i.e. false, null, true [72, 71]. But JSON schema e.g. for IFC4 specification defined by BIMserver [70] as shown in Table 1-Right uses customized types i.e. "type": "IfcPersonAndOrganization" which does not conform to JSON schema specification although it can be parsed and used within JavaScript applications.

Secondly, this schema does not address mapping of required and optional attributes, for instance in the example shown in Table 2-Left which is extracted from the BIMserver example files [69], “The Person” and “The Organization” attributes cannot be null according to IFC specification but it is considered as an empty object which is incorrect, because BIMserver schema definition [70] shown in Table 2-Right does not consider the constraints of optional/required attributes as defined in IFC specification.

Thirdly, IFC schema definition based on JSON serialization in BIMserver [70] is incomplete, for instance, enumeration types are not defined. In IFC4 specification "IfcOwnerHistory" the attribute "State" is of type "IfcStateEnum" with enumerated values of "READWRITE", "READONLY", "LOCKED", "READWRITELOCKED", and "READONLYLOCKED" but as shown in Table 2-right for "State" attribute in addition to the incorrect definition of "type" value as discussed above, the enumeration values are not specified and "IfcStateEnum" together with all enumeration types are considered as empty values.

Table 2 Definition of JSON document and JSON schema in BIMserver

BIMserver JSON document

```
"5374396": {
    "Owner History": {
        "Owning User": {
            "The Person": {},
            "The Organization": {}
        },
        "Owning Application": {},
        "State": "NULL",
        "Change Action": "NOCHANGE",
        "Creation Date": 1215091190
    },
    "Tag": "BF42B38D-335C-4620-80-93-9FFDAC5BDD4C"
},
```

Parts of BIMserver JSON Schema

```
"IfcOwnerHistory": {
  "domain": "ifcutilityresource",
  "superclasses": [],
  "fields": {
    "OwningUser": {
      "type": "IfcPersonAndOrganization",
      "reference": true,
      "many": false
    },
    "OwningApplication": {
      "type": "IfcApplication",
      "reference": true,
      "many": false
    },
    "State": {
      "type": "enum",
      "reference": false,
      "many": false
    },
```

Another JSON based schema definition used in solutions like bimJSON [73] is inspired by GeoJSON. The bimJSON itself is still in development. GeoJSON is a JSON format for encoding geographic data structures and supports some geometry types such as Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon [74, 75]. The new and recently published standard specification of the GeoJSON format is RFC 7946 [75]. In GeoJSON, "Features" contain a Geometry object and additional properties, and a

"FeatureCollection" contains a list of Features. The value of the "properties" keyword is a JSON object that contains any attribute other than the geometry data, a simple example is shown in Figure 21.

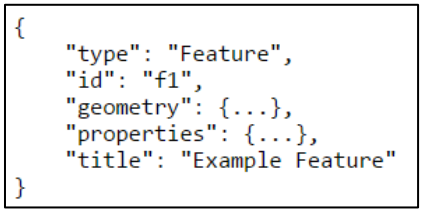


Figure 21 Example of a GeoJSON "Feature" [75]

Although GeoJSON is an internet standard, so far there is no published schema definition for it and most importantly, there is no GeoJSON schema for the building specific data. In fact, GeoJSON does not aim to address the need of the AEC industry. GeoJSON is not suitable for a well-defined building data schema because firstly, since it is concerned with geographic data in the broadest sense [75] it does not support complex geometry definition and secondly, since it only aims at serializing geographic data, other data than geometry definition should be specified within "properties" keyword and "properties" definition is too broad with no specification to address building data components and attributes like it is specified in IFC schema. IFC schema on the other hand, contains both geometry and non-geometry properties of the building data [20]. This study introduces a methodology to generate a standardized JSON encoding of IFC specification that is based on both IFC specification [60] and JSON schema [71].

4.3 Developing ifcJSON

The ifcJSON representation introduced in this paper is the JSON implementation of IFC data schema (i.e. in an EXPRESS schema specification). This study implements ifcJSON4 which is based on IFC4 Add2 specification i.e. current IFC release. In addition, the ifcJSON4 schema developed in this study follows the structure of recent JSON Schema definition published by Internet Engineering Task Force (IETF) [71]. This chapter first introduces the implementation approach and data model mapping for implementing ifcJSON4 schema and documents. Then, it evaluates the implementation in a use case approach.

4.3.1 Methods for JSON Encoding of IFC Specification

To implement JSON encoding of IFC specification there are two major methodologies that can be applied: 1) to translate ifcXML to JSON serialization, or 2) to directly convert IFC EXPRESS specification to ifcJSON schema.

The first method is to translate ifcXML specification to JSON serialization using XML to JSON translating algorithms. However, as Wang has pointed out [47], in application development there is still a problem in data transmission between XML and JSON serialization in a correct and highly-efficient way. The analysis of original XML data structure and translated JSON data structure shows that although the translation of data between XML and JSON can be applied, data mismatches will be caused and the data accuracy will decrease [47]. A quick experiment within the direct translation of ifcXML schema to JSON schema using XML to JSON translators indicated a lot of redundant and duplicated data. The solution to this problem is to develop an improved algorithm or to manually fix “name/value” pairs in JSON serialization [47]. Another issue in translating

ifcXML to JSON is that ifcXML schema implementation itself has limitations: “If the ifcXML schema is used to generate a model in a database implementation that is not compliant to the EXPRESS standard then certain conflicts or limitations may be encountered” [58] and the list of limitations as stated in the ifcXML implementation guide [58] cannot be definitive. The methodology of translating ifcXML to ifcJSON is not accurate and effective, so it is not recommended.

The second method is to directly translate IFC EXPRESS to JSON serialization. However, there is no standardization on configuration of mapping between EXPRESS and JSON and there is no available methodology for automatic conversion between IFC EXPRESS and ifcJSON. Therefore, this study had to develop the methodology for implementing ifcJSON schema definition from IFC EXPRESS specification.

4.3.2 ifcJSON4 Implementation Approach

Unlike ifcXML schema specification that is an automatic conversion from the EXPRESS (ISO 10303 part 1) representation of the IFC schema, there is no available methodology for automatic conversion between IFC EXPRESS and ifcJSON. In addition, there is no standardization on configuration of mapping between EXPRESS and JSON in general. Therefore, in this study first a methodology to generate ifcJSON schema is developed that is interpreted from the IFC specification [60] and is constrained by the JSON schema [71]. This conversion needs to be done based on the in-depth understanding of both EXPRESS and JSON data representation. Then, the chapter describes the data content of an ifcJSON document and its data validation process. The methodology for data model mapping in this chapter (shown in Figure 22) can be summarized in three main steps:

Figure 22 Methodology for ifcJSON Data Model Mapping

a) ifcJSON4 Schema development, which is the JSON schema definition that corresponds to the IFC4 EXPRESS definition. This is developed through the generation of a schema from the IFC4 EXPRESS source definition as well as JSON Schema specification defined by IETF [71].

b) ifcJSON document implementation, which is a JSON document that can be validated against the ifcJSON4 schema. Each ifcJSON document should be well structured and validated for formatting JSON data and it should also pass the validation against ifcJSON4 schema.

c) Data validation approach, which addresses three validation approaches including the syntactic validation of ifcJSON data, the validation of ifcJSON4 Schema against the original JSON Schema, and the validation of ifcJSON document data content against the ifcJSON4 schema.

JSON Schema is an “Internet Draft” which latest draft is published in October 2016 [71]. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF) and can be referred as a “work in progress” [71]. JSON Schema itself is written in JSON which specifies a JSON-based format for defining and validating the structure of JSON data. JSON Schema "is a JSON media type for defining the structure of JSON data. JSON Schema is intended to define validation, documentation, hyperlink navigation, and

interaction control of JSON data" [71]. So, this study uses this specification of JSON schema in addition to IFC4 Add2 specification to develop ifcJSON4 which, similar to JSON schema, has a JSON-based format itself. In addition, since one of the purposes of JSON Schema is instance validation that is described by IETF [71] the study then validates the ifcJSON4 schema against the original JSON schema version 4 by IETF. Additionally, this study specifies the data mapping approach for ifcJSON document data content that can address the requirements of ifcJSON4 Schema. The content of the ifcJSON document described here, can be represented in ".ifcjson" or ".json" format which is the exchange format for ifcJSON. This document should be validated both with a validator for formatting JSON data as well as against the ifcJSON4 schema.

In addition to providing a data model mapping for ifcJSON, the proposed implementation is applied in a use case to evaluate the data content of ifcJSON exchange model. There are three fundamental entity types in the IFC model as object definitions, relationships and property definitions and this chapter focuses on the representation of IFC object definition. The use case selected in this study indicates the implementation of ifcJSON representation for object definition in precast concrete exchange model known as EMPC.1. Precast Concrete BIM standard [76] defines the specification of the Model View Definitions for twelve precast model exchanges [77] and specifies EMPC.1 as a subset of IFC schema. Since the Precast Concrete BIM standard project is still an ongoing development, this study applies the implementation of ifcJSON in precast concrete exchange model as a use case. Building Information Modeling Standard for Precast Concrete Construction specifies EMPC.1 exchange model [76] that consists of architectural concept model or engineering concept model passed to detailer for further

preliminary precast structural and fabrication detailing. The use case approach here generates an ifcJSON document that contains three information groups, known as IFC concepts listed below to represent IFC object definition.

- Owner History Data: While owner history data is optional in IFC schema [60], EMPC.1 requires history and identification data as exists in IfcOwnerHistory entity (Figure 23) to be mandatory in the exchange model.

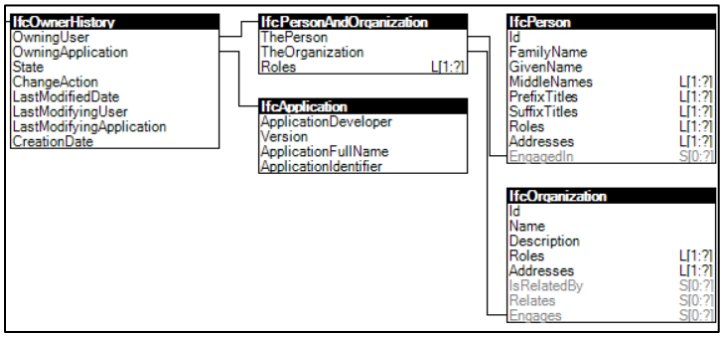


Figure 23 IfcOwnerHistory in IFC4 EXPRESS specification

- Geometry Representation Data: EMPC.1 requires the model to use extruded geometry for representation of the building elements. Therefore, Body SweptSolid Geometry concept definition (Figure 24) which is specified in IFC4 schema [60] is applied in the use case.

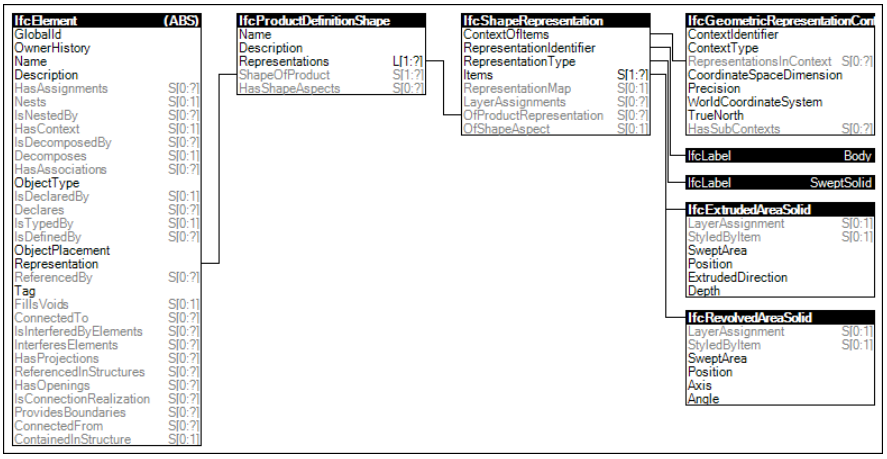


Figure 24 Instance Diagram in IFC4 EXPRESS specification for Body SweptSolid Geometry

- Product Placement Data: Product occurrences can be placed in 3D space relative to where they are contained. IFC4 concept definition for product placement is illustrated in Figure 25. For building elements positioning is relative to the containing spatial structure. If a containing spatial structure contains a grid, then placement may also be based relative to grid coordinates [60]. In IFC schema, ObjectPlacement attribute is optional but in EMPC.1 the provision of product placement data of precast pieces is required.

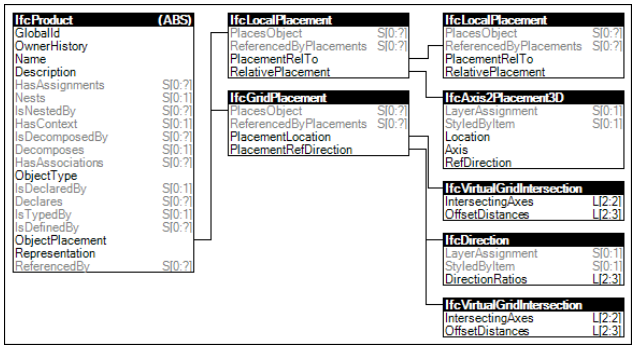


Figure 25 Instance diagram in IFC4 EXPRESS specification for Product Placement concept

4.3.3 Implementation Challenges and Limitations

The implementation of ifcJSON faces several challenges and issues listed below.

- There is a lack of standardization on translating EXPRESS model to JSON data. There is no automated methodology, standard or guideline to guide mapping from EXPRESS to JSON.
- There is no documentation on a methodology to convert the IFC source definition in EXPRESS into a JSON schema.
- The validation of ifcJSON document needs to ensure that the document is well formed and that it conforms to ifcJSON4 schema. Currently there is no tool that can handle these two validation methods at the same time. Therefore, these two validation approaches need to be done separately.
- There is no tool for visualizing. ifcjson file in terms of the geometry representation and the attributes. Therefore, after generating the ifcJSON document and validating it, this document can be translated back to an SPF format in order to visualize model in model viewers (e.g. Solibri).

Besides, some of the limitations of the proposed ifcJSON implementation can be summarized as follows. Three fundamental entity types in the IFC model are object definitions, relationships and property definitions and this chapter only specifies JSON representation of IFC object definition. In addition, the ifcJSON4 schema that is generated based on the IFC4 EXPRESS schema, is not the IFC schema as a whole. It is only generated with a focus on some limited entities needed for the use case in this study. IFC EXPRESS specification includes data item for types, entities, rules and functions [60] but ifcJSON

implementation here is similar to the schema derivation logic in ifcXML implementation [58] and hence loses some constraints including rules, inverse relationships and derived attributes. Therefore, ifcJSON only implements IFC types and entities. In addition, some of the supertype/subtype dependencies [57] are summarized in a way that it can manage limited number of entities to be implemented. For instance, all the inherited attributes of the supertype entities are considered in the subtype as if the supertype exists but the supertype entity might not be shown. Moreover, some of the EXPRESS datatypes such as defined datatypes and select datatypes [57] have been reduced in accordance with the use case in this study to facilitate the generation of the ifcJSON schema.

4.4 Data Model Mapping

JSON Schema [71], in general, defines a structure that governs the JSON data and defines the validation and interaction control of JSON data [71]. Objects are the mapping type in JSON. In fact, they map “keys” to “values”. The “keys” must be defined as strings (e.g. “title” or “description”). Also, there are seven primitive types for JSON values including array, boolean, integer, number, null, object, and string. Each of these key-value pairs is referred to as a “property” or sometimes it might be called as a “member” [71]. On the other hand, a JSON Schema is a JSON document itself. This document must be an object. Object properties (or members) defined by JSON Schema are known as schema keywords such as "required". A JSON Schema may contain properties which are not schema keywords [71]. An example is shown in Figure 26.

Figure 26 An example of a JSON Schema

4.4.1 ifcJSON4 Schema

The ifcJSON4 schema implemented here has seven major properties (i.e. keywords) shown in Table 3 which follows the structure of JSON Schema [71] and uses the latest published draft v6 specification [78, 79].

Table 3 JSON schema main keywords in ifcJSON4

ifcJSON4 Schema Keywords	**Description**
"$schema"	Declares that this JSON document is a piece of JSON Schema and states that this schema is written according to the draft v6 specification.
"title"	States that this schema describes a product with the title "ifcJSON4 schema". It does not add constraints to the data validation.
"description"	The intent of the schema is stated here with more detail. It does not add constraints to the data validation.
"definitions"	It is a standardized placeholder in which inline subschemas can be defined to be used in a schema. It includes select datatypes in IFC schema. IFC simple and enumeration datatypes are not included within this subschema as will be explained later.

"type"	Defines that ifcJSON data must be a JSON Object which is considered as a constraint.
"properties"	Includes all the IFC entities with a set of attributes for each entity.
"required"	Indicates what IFC entities are required – if at all- (based on the IFC specification) in a JSON document. Two examples of required entities are shown.

In addition, the "$ref" keyword is the JSON reference [78, 79], which here is local to the schema, and the fragment part is a Unique Reference Identifier (URI) encoded JSON pointer to the address property of the definitions keyword. It is used in defining the dependencies of IFC entities. A brief simple representation of ifcJSON4 schema is shown in Table 4. As explained in section 4.4, ifcJSON4 schema here describes IFC types and entities. In ifcJSON4 schema root, IFC types are shown under "definitions" and IFC entities are represented under "properties" as illustrated in Table 4 which only shows a set of examples of what are included under "definitions" and "properties" schema keywords. These types and entities are specified as empty objects for now in this table for representation purposes and will be described in details in the following sections.

Table 4 Brief representation of the overall ifcJSON4 Schema structure

ifcJSON4 Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "ifcJSON4 Schema",
  "description": "This is the schema for representing IFC4 data in JSON",
  "definitions": {
    "description": {"_comment": "This includes select IFC datatypes."},
    "ifcValue": {},
    " ifcAxis2Placement": {}
  },
  "type": "object",
  "properties": {
    "description": {"_comment": "This includes all IFC entities"},
    "ifcOwnerHistory": {},
    "ifcProject": {}
    "ifcCartesianPoint": {},
    "ifcAxis2Placement2D": {},
    "ifcAxis2Placement3D": {}
  },
  "required": ["ifcOwnerHistory", "ifcProject"]
}
```

It needs to be noted that in JavaScript, anything that is not a constructor usually starts with lowercase and in "camelCase" convention. Since the ifcJSON4 schema is going to be parsed mostly in JavaScript applications, this study uses the "camelCase" convention to name the properties and definitions (i.e. IFC entities and types).

As explained above, ifcJSON root represents IFC types under "definitions" and IFC entities under "properties" keywords. In ifcJSON4 schema the "definition" keyword performs as a subschema [79] and lists select datatypes such as IfcAxis2Placement. In IFC EXPRESS schema, the IfcAxis2Placement type is a select type that can be either IfcAxis2Placement2D entity or IfcAxisPlacement3D entity. Therefore, in ifcJSON4 schema under "definitions", the IfcAxis2Placement type can be defined as shown in Table 5 which refers to either ifcAxis2Placement2D or ifcAxis2Placement3D entity. Therefore, instead of using the IfcAxis2Placement type, this representation can be used to refer to IfcAxis2Placement2D or IfcAxisPlacement3D.

Table 5 IfcAxis2Placement in IFCJSON4 schema

ifcJSON4 Schema

`"ifcAxis2Placement": {`
`  "type": "object",`
`  "oneOf":[`
`  { "$ref": "#/properties/ifcAxis2Placement2D" },`
`  { "$ref": "#/properties/ifcAxis2Placement3D" } ]}`

Since IfcAxis2Placement2D and IfcAxis2Placement3D are main IFC entities, these are defined under "properties" in the ifcJSON4 schema root. The ifcJSON representation of these two entities are shown in Table 6 which lists all the attributes based on IFC specification.

Additionally, in this subschema simple IFC datatypes are not included because they can be directly applied in JSON. IFC base types like IfcIdentifier or IfcText can be defined as a type or a set of types directly in JSON. This simplifies the ifcJSON4 schema. For instance, IfcProperty in IFC EXPRESS schema, as shown in Table 7-Left has an attribute "Name" that is defined as "IfcIdentifier" in IFC EXPRESS schema. On the other hand, "IfcIdentifier" type is defined as a string with 255 character length. Therefore, instead of implementing the "IfcIdentifier" as a separate type, the "Name" attribute can be defined in ifcJSON4 as a property which type is string with maximum length of 255 characters as shown in Table 7-Right.

Table 6 IfcAxis2Placement2D and IfcAxis2Placement3D entities in ifcJSON4 schema

ifcJSON4 Schema

```
{
 "ifcAxis2Placement2D": {
       "type": "object",
       "properties": {
              "instanceId": { "type": "integer" },
              "ifc": { "type": "string" },
              "location": {
                     "type": "object",
                     "allOf": [{ "$ref": "#/properties/ifcCartesianPoint" }]
              },
              "refDirection": {
                     "oneOf":[
                            { "type": "null" },
                            {  "type": "object",
                                   "allOf": [
                                          {  "$ref":  "#/properties/ifcDirection"
                                          }] }]
              }
       },
       "required": [ "location", "refDirection"]
},
"ifcAxis2Placement3D": {
       "type": "object",
       "properties": {
              "instanceId": { "type": "integer" },
              "ifc": { "type": "string" },
              "location": {
                     "type": "object",
                     "allOf": [{ "$ref": "#/properties/ifcCartesianPoint" }]
              },
```

```
            "axis": {"type" : ["object", "null"]},
            "refDirection": {
                  "oneOf":[
                        { "type": "null" },
                        {  "type": "object",
                        "allOf": [{ "$ref": "#/properties/ifcDirection" }]
                  }]
            }
      },
      "required": ["location", "axis", "refDirection"]
 }
}
```

Table 7 IfcProperty and the IfcIdentifier type entity, Right: IfcProperty entity in ifcJSON4 schema

IFC EXPRESS Schema	ifcJSON4 Schema

ENTITY IfcProperty
…
Name : IfcIdentifier;
…
END_ENTITY;
 	
TYPE IfcIdentifier = STRING (255);
END_TYPE;
	"ifcProperty": {
	"type": "object",
	"properties": {
	"ifc": { "type": "string" },
	"name": {
	"type": "string",
	"maxLength": 255},
	"description": {
	"type": ["string", "null"]}
	},
	"required": ["name", "description"]}

Also, IFC enumeration types can be applied directly in JSON schema. For instance, in EXPRESS definition of IfcRectangleProfileDef, the attribute “ProfileType” is defined as IfcProfileTypeEnum and then IfcProfileTypeEnum is defined as an enumeration type of “CURVE” and “AREA”. In ifcJSON4 schema, the property “profileType” can be defined directly as an enumeration of “CURVE” and “AREA” shown in Table 8 without the need for implementing the ifcProfileTypeEnum type entity. This simplifies the ifcJSON4 schema.

Table 8 “ProfileType” attribute in ifcJSON4 schema

ifcJSON4 Schema
`"profileType": { "enum": ["CURVE", "AREA"]}`

In IFC specification [60] class of information i.e. IFC entities defined by common attributes and constraints. In ifcJSON schema, each IFC entity has a “properties” keyword that lists its attributes following JSON schema convention [71, 78, 79] as shown in Table 6 and Table 7-Right.

In IFC specification two types of constraints are applied on attributes [60]. The first constraint which is the most general constraint is about the existence of attribute values: mandatory and optional attributes. Values of mandatory attributes must be provided while values of optional attributes can be assigned as null. In EXPRESS, an optional attribute is shown by the word “OPTIONAL” before the attribute name [57]. Such attribute is not required to have a value in a given entity instance. In the IFC EXPRESS specification [60], since the intention is to cover many exchange use cases at different stages of the building life cycle, there are many attributes that are defined as optional. For instance, the EXPRESS definition of IfcAxis2Placement3D, defines the value of “axis” and “refDirection” attributes as optional. In EXPRESS, when an entity instance defines an explicit value for its optional attribute, it should be in accordance with the datatype of the attribute. But when in an entity instance, the value for its optional attributes is not supplied, the attribute still occurs in the entity instance but the value for this attribute should be shown as the dollar sign "$" [59].

On the other hand, in the JSON schema, value of the keyword “required” is an array with at least one element which elements are unique strings [79]. To implement the notion of EXPRESS optional attribute in ifcJSON4 schema, all properties need to be considered as “required” to specify that the value of this attribute must occur in the entity instance either with an explicit value or as null. Additionally, the properties which values are optional in the IFC specification, should be defined in their type (i.e. "type": ["object", "null"]) to let the property accept null values. Therefore, IfcAxis2Placement3D in ifcJSON4 schema is defined as shown in Table 9 to represent “axis” and “refDirection” can accept null values.

Table 9 IfcAxis2Placement3D in ifcJSON4 schema

ifcJSON4 Schema

```
{
  "type": "object",
  "properties": {
       "instanceId": { "type": "integer" },
        "ifc": { "type": "string" },
       "location": {
         "type": "object",
         "allOf": [{ "$ref": "#/properties/ifcCartesianPoint" }] },
       "axis": {"type" : ["object", "null"]},
       "refDirection": {
         "oneOf":[
              { "type": "null" },
              {  "type": "object",
              "allOf": [{ "$ref": "#/properties/ifcDirection"}] }
         ]}
       },
       "required": ["location", "axis", "refDirection"]
}
```

Table 10 IfcDirection in ifcJSON4 schema

ifcJSON4 Schema

```
{
        "type": "object",
        "properties": {
                "instanceId": { "type": "integer" },
                 "ifc": { "type": "string" },
                "directionRatios": {
                        "type": "array",
                        "items": {"type": "number"},
                        "minItems": 2,
                        "maxItems": 3
                }
        },
        "required": ["directionRatios"]
}
```

The second constraint that is applied on attributes in IFC specification is for aggregation data types such as Set, List, or Array, known as the existence constraint or cardinality constraints which is often defined by a minimal and maximal number of elements [60]. In ifcJSON4 schema for collection-based attributes (e.g. LIST or SET), data existence and cardinality is defined directly by JSON validation keywords for arrays such as “minItems” (i.e. must be greater than, or equal to, 0) or “maxItems” (i.e. must be greater than, or equal to, 0). For instance, EXPRESS definition of IfcDirection specifies the “directionRatios” attribute as a collection attribute that must contain at least two members and maximum of three members. This cardinality is defined directly in ifcJSON4 schema, indicated in Table 10. The “directionRatios” in the schema indicates that this is an array of numbers with minimum 2 items and maximum 3 items.

4.4.2 ifcJSON documents

An ifcJSON document is the exchange data for ifcJSON and should be validated against ifcJSON4 schema. The ifcJSON document consists of objects or an array of objects that represent IFC entity instances and its structure is defined by ifcJSON4 schema. Each

data element that can be exchanged within an SPF file, can also be exchanged within an ifcJSON document. For instance, ifcJSON4 schema (Appendix 1) can be used to test the validity of the ifcJSON data shown in Table 11-Left. This JSON data can be mapped to the representation in the SPF file shown also in Table 11-Right.

Table 11 An example of ifcAxis2Placement2D in ifcJSON document (Left), An example of ifcAxis2Placement2D entity in an SPF file (Right)

ifcJSON document

```
{
       "instanceId": 107,
       "ifc": "IFCAXIS2PLACEMENT2D",
       "location": {
              "instanceId": 9,
              "ifc": "IFCCARTESIANPOINT",
              "coordinates": [0, 0]
       },
       "refDirection": {
              "instanceId": 29,
              "ifc": "IFCDIRECTION",
              "directionRatios": [0, -1]
       }
}
```

IFC SPF format data

```
#9=
IFCCARTESIANPOINT((0.,0.));
#29= IFCDIRECTION((0.,-1.));
#107=
IFCAXIS2PLACEMENT2D(#9,#29);
```

In EXPRESS, when encoding an SPF file, each entity instances are represented by an entity name which is encoded as a number sign "#" followed by a sequence of digit characters represented as a number [59]. In this ifcJSON4 schema, “instanceId” property is defined to keep track of the instance references. As an example, the schema for an IfcCartesianPoint is shown in Table 12 indicating the type for its “instanceId”. As explained earlier, there might be a need for converting ifcJSON documents to SPF file for validation or visualization purposes. In that case, including “instanceId” facilitates the translation of JSON document to an SPF file. The “instanceId” property can be considered as a required property if the JSON schema is decided to mandate the provision of instanceIds later. For now, it is considered as optional.

Table 12 IfcCartesianPoint in ifcJSON4 schema

ifcJSON4 Schema

```
{
        "type": "object",
        "properties": {
                "instanceId": { "type": "integer" },
                "ifc": { "type": "string" },
                "coordinates": {
                        "type": "array",
                        "items": {"type": "number"},
                        "minItems": 1,
                        "maxItems": 3}
        },
        "required": ["coordinates"]
}
```

In ifcXML specification, the attribute “Id” is used as XLink to create hyperlinks within XML documents [58]. The specifications for XML via XLink has hypertext references built into their respective specifications. In JSON Schema, the "id" keyword defines a URI for the schema and subschemas can use "id" too to give themselves a document-local identifier [71]. In ifcJSON schema developed here, the “instanceId” keyword refers to the entity instances of the SPF file to provide the capabilities of translating JSON document to SPF file. In this study, the “instanceId” is considered as an integer data type.

4.4.3 Data Validation Methodology

The ifcJSON documents developed here, should be proved to be complete and well-formed. Therefore, this document needs to be validated regarding two aspects: firstly, to check the syntactic constraint based on JSON data formatting and secondly, for structural constraints defined in ifcJSON4 schema. Moreover, the ifcJSON4 schema should be proven to be well-formed and complete too and thus, needs to be validated both for syntactic conformity and against the JSON schema [71]. Most importantly, ifcJSON

schema should also be defined correctly when comparing with the IFC specification, which should be done manually following the methodology explained for ifcJSON4 schema implementation presented earlier.

To validate ifcJSON data for formatting, this study uses a JSON validator (e.g. http://jsonlint.com/) to validate the JSON data for correctness. This helps debugging JSON data and reports failure if the document is not well-formed due to incorrect JSON syntax. This validation technique should be applied to JSON document and to ifcJSON4 schema which is a JSON document itself.

JSON schema is used to validate the structure and data types of a piece of JSON document [71]. To validate the ifcJSON4 schema against current JSON schema [71, 78], two approaches can be used. This study uses the "JSON Schema Lint" [80] which is a JSON schema validator to help write and test any JSON Schema that conforms to the JSON schema draft v6 specification. An alternative approach is to use the methodology similar to validating ifcJSON document against the ifcJSON4 schema. Since the ifcJSON4 Schema is an ifcJSON document itself, this methodology can be used to validate the ifcJSON4 schema against the main JSON schema draft V6. Here, both these approaches are applied for the schema validation.

For validating the ifcJSON documents against the ifcJSON4 schema, JSON Schema validators should be used. There are some implementations of JSON Schema validators with specific functionalities. In this study, "ajv" is used which is licensed by MIT and is available on GitHub that is an open source JSON schema validator for both

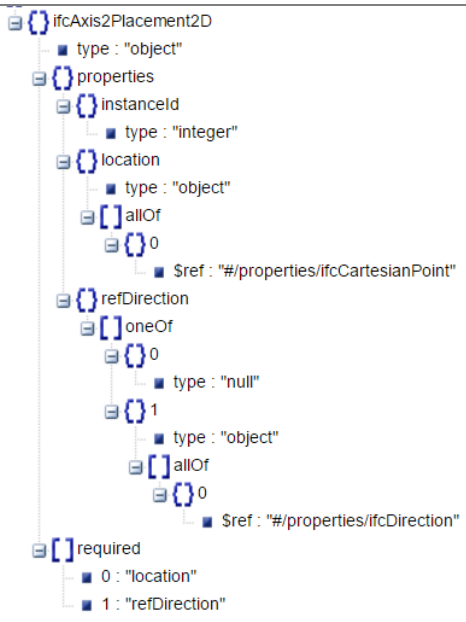
node.js and browser. It implements full JSON schema draft V6 standard. ajv generates code to turn JSON schemas into JavaScript functions [81].

In the ifcJSON document shown in Table 13-Left, the "location.coordinates" property is missing. According to ifcJSON4 schema in Table 13-Right, this document should return an error when validated against the schema.

Therefore, when the document in Table 13-Left is validated against ifcJSON4 schema illustrated in Table 13-Right, the validator should report an error. The error reporting of ajv when using a browser is shown in Figure 27 that specifies where the error occurs. When the data is complete, for instance when the “Coordinates” property is added to the same example, the report shows a successful validation.

Table 13 An example of IfcAxis2Placement2D in ifcJSON document with a missing property (Left), ifcJSON4 schema representation for IfcAxis2Placement2D (Right)

ifcJSON document	**ifcJSON4 Schema**

<pre>{
"instanceId": 107,
"ifc": "IFCAXIS2PLACEMENT2D",
"location": {
"instanceId": 9,
"ifc": "IFCCARTESIANPOINT",
},
"refDirection": {
"instanceId": 29,
"ifc": "IFCDIRECTION",
"directionRatios": [0, -1]
}
}</pre>
	

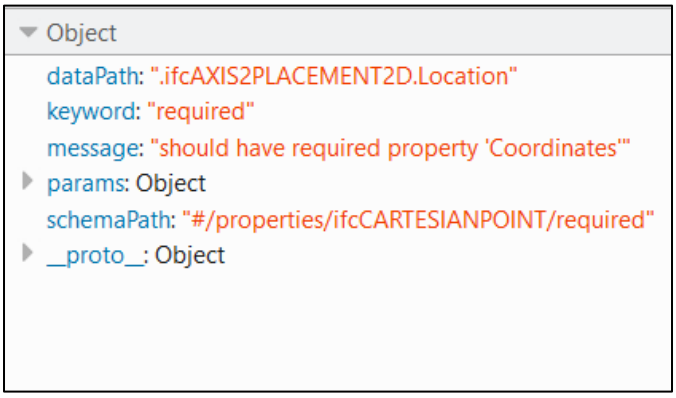


Figure 27 Browser error reporting of ajv

4.5 Use Case Approach

This section explains how to apply ifcJSON4 schema in a use case. As mentioned earlier, the use case in this study is the exchange model EMPC.1 [76] in the precast concrete BIM Standard. More specifically, a precast concrete column is used shown in Figure 28. This exchange model, consists of concept design layout of precast pieces. The exchange happens at the preliminary stage of the project when different disciplines such as architecture, structural engineering, and precast engineering is exchanging the model for conceptual design purposes. The instance model in this study consists of a single column within an architectural or engineering concept model which is passed to the detailer for further preliminary precast structural and fabrication detailing.

ifcJSON4 schema and the ifcJSON document for this exchange model is described here for four data categories that indicate the geometry data representation (Figure 24),

object placement data (Figure 25), owner history data (Figure 23) and the building element data for a precast concrete column that includes these three data categories. The schema is represented in Appendix 1.

Name	**Concrete Column**
Profile Type	Rectangle
Geometry	Extrusion
X Dim	0.45
Y Dim	0.30
Length	3.00
Top Elevation	3.00
Bottom Elevation	0.00
Global X	0.00
Global Y	0.00

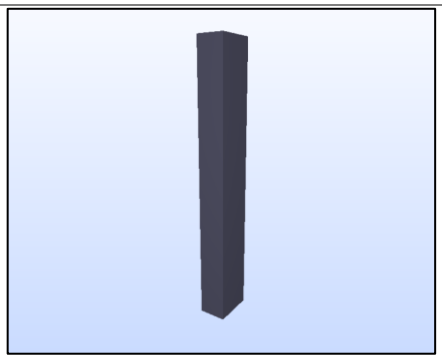


Figure 28 A precast concrete column with its features and dimensions (metric)

4.5.1 Geometry Representation Data

The geometry representation for this use case is an extruded geometry or "IfcExtrudedAreaSolid". Therefore, for the ease of schema representation here, the geometry representation entities in the ifcJSON4 schema (Appendix 1) have been limited to the entities required for the use case here. In IFC EXPRESS schema, IfcProductDefinitionShape.Representations is a list of IfcRepresentation entities. Besides, IfcRepresentation could be one of IfcShapeModel or IfcStyleModel which are narrowed down to IfcShapeModel. Accordingly, IfcShapeModel can be either IfcShapeRepresentation or IfcTopologyRepresentation which here is narrowed down to IfcShapeRepresentation. Moreover, here the IfcShapeRepresentation.RepresentationType is only considered as "Swept Solid" while in IFC4 EXPRESS schema SolidModel includes Swept solid, Boolean results and Boundary Representation (B-rep) bodies with more

specific types as: SweptSolid, AdvancedSweptSolid, Brep, AdvancedBrep, CSG, and Clipping.

In addition, in IFC4 EXPRESS specification IfcShapeRepresentation.Items is a set of IfcRepresentationItem but IfcRepresentationItem here is narrowed down to only IfcGeometricRepresentationItem. Accordingly, IfcGeometricRepresentationItem is narrowed down to IfcSolidModel, IfcSolidModel is narrowed down to IfcSweptAreaSolid, and IfcSweptAreaSolid is narrowed down to IfcExtrudedAreaSolid which is the entity used in the schema (Appendix 1) as IfcShapeRepresentation.Items. In the schema definition for this use case (Appendix 1), the abstract super-type and subclass have been merged while all the inherited attributes (i.e. "SweptArea" and "Position") are included in the IfcExtrudedAreaSolid entity. Also, the data type for IfcPositiveLengthMeasure is applied directly in this JSON schema, shown in defining "Depth" property.

Table 14 IfcProductDefinitionShape in an SPF file

IFC SPF format data
`#6= IFCCARTESIANPOINT((0.,0.,0.));`
`#9= IFCCARTESIANPOINT((0.,0.));`
`#29= IFCDIRECTION((0.,-1.));`
`#19= IFCDIRECTION((0.,0.,1.));`
`#31= IFCAXIS2PLACEMENT3D(#6,$,$);`
`#107= IFCAXIS2PLACEMENT2D(#9,#29);`
`#108= IFCRECTANGLEPROFILEDEF(.AREA.,$,#107,0.4572,0.3048);`
`#110= IFCEXTRUDEDAREASOLID(#108,#31,#19,3.048);`
`#111= IFCSHAPEREPRESENTATION(#67,'Body','SweptSolid',(#110));`
`#252= IFCPRODUCTDEFINITIONSHAPE($,'DetailedProfile',(#111));`

The IFC instances as part of an SPF file shown in Table 14, indicate the geometry shape representation by defining an IfcProductDefinitionShape entity with the shape relevant information. Table 15-left shows the data related to IfcExtrudedAreaSolid in an

ifcJSON document that passes the validation against the ifcJSON4 schema in Appendix 1. ifcJSON4 schema for IfcExtrudedAreaSolid is visualized in Table 15-Right. In the schema definition for this use case, the abstract super-type and subclass have been merged while all the inherited attributes (i.e. “SweptArea” and “Position”) are included in the IfcExtrudedAreaSolid entity. In addition, in IFC EXPRESS specification IfcPositiveLengthMeasure is defined as IfcLengthMeasure which is a simple IFC type [60]. As mentioned, the data type for IfcPositiveLengthMeasure is applied directly in ifcJSON4 schema, shown in defining “Depth” property.

Table 15 IfcExtrudedAreaSolid in ifcJSON document (Left), ifcJSON4 schema representation for IfcExtrudedAreaSolid (Right)

ifcJSON document

```
{
    "instanceId": 110,
    "ifc": "IFCEXTRUDEDAREASOLID"
    "sweptArea": {
        "instanceId": 108,
        "ifc": "IFCRECTANGLEPROFILEDEF"
        "profileType": "AREA",
        "profileName": null,
        "position": {
            "instanceId": 107,
            "ifc": "IFCAXIS2PLACEMENT2D"
            "location": {
                "instanceId": 9,
                "ifc": "IFCCARTESIANPOINT"
                "coordinates": [0, 0] },
            "refDirection": {
                "instanceId": 29,
                "ifc": "IFCDIRECTION"
                "directionRatios": [0, -1] }
        },
        "xDim": 0.45,
        "yDim": 0.30
    },
    "position": {
        "instanceId": 31,
        "ifc": "IFCAXIS2PLACEMENT3D"
        "location": {
            "instanceId": 6,
            "ifc": "IFCCARTESIANPOINT"
```

ifcJSON4 Schema

```
"ifcExtrudedAreaSolid": {
  "type": "object",
  "properties": {
    "instanceId": {
      "type": "integer"
    },
    "ifc": {
      "type": "string"
    },
    "sweptArea": {
      "type": "object",
      "allOf": [
        {
          "$ref": "#/properties/ifcRectangleProfileDef"
        }
      ]
    },
    "position": {
      "oneOf": [
        {
          "type": "null"
        },
        {
          "type": "object",
          "allOf": [
            {
              "$ref": "#/properties/ifcAxis2Placement3D"
            }
          ]
        }
      ]
    },
    "extrudedDirection": {
      "type": "object",
      "allOf": [
```

```
            "coordinates": [0, 0, 0]
        },
        "axis": null,
        "refDirection": null
    },
    "extrudedDirection": {
        "instanceId": 19,
        "ifc": "IFCDIRECTION"
        "directionRatios": [0, 0, 1]
    },
    "depth": 3.0
}
```

```
            {
              "$ref": "#/properties/ifcDirection"
            }
          ]
        },
        "depth": {
          "type": "number",
          "minimum": 0,
          "exclusiveMinimum": false
        }
      },
      "required": [
        "sweptArea",
        "position",
        "extrudedDirection",
        "depth"
      ]
    }
```


Moreover, the geometry data for IfcProductDefinitionShape represented in an ifcJSON document is shown in Table 16-Left and passes the validation against the schema illustrated in Table 16-Right. As mentioned, the RepresentationType for IfcShapeRepresentation is considered as “Swept Solid” in the ifcJSON document. This ifcJSON document reflects the same data content for geometry representation as IFC SPF file shown in Table 14 and passes the validation against the ifcJSON4 Schema. Here, the data defined as “Representations.Items” in ifcProductDefinitionShape shows IfcExtrudedAreaSolid data which is represented in Table 15-Left.

Table 16 IfcProductDefinitionShape in ifcJSON document (Left), ifcJSON4 schema representation for IfcProductDefinitionShape (Rright)

ifcJSON document

```
{
    "instanceId": 252,
    "ifc": "PRODUCTDEFINITIONSHAPE",
    "name": null,
    "description": "DetailedProfile",
    "representations": [{
      "instanceId": 111,
      "ifc": "IFCSHAPEREPRESENTATION",
      "contextOfItems": {
         "instanceId": 67,
         "ifc": "IFCGEOMETRICREPRESENTATIONCONTEXT",
         "contextIdentifier": null,
         "contextType": "Model",
         "coordinateSpaceDimension": 3,
         "precision": 1.0,
         "worldCoordinateSystem": {
             "instanceId": 64,
             "ifc": "IFCAXIS2PLACEMENT3D",
             "location": {
                 "instanceId": 6,
                 "ifc": "IFCCARTESIANPOINT",
                 "Coordinates":[0,0,0]
             },
             "axis": null,
             "refDirection": null
         },
         "trueNorth": {
           "instanceId": 65,
           "ifc": "IFCDIRECTION",
           "directionRatios":[2,6.1, 1]
         }
      },
      "representationIdentifier": "Body",
      "representationType": "SweptSolid",
      "items": [{// see Table 15 }]
    }]
}
```

ifcJSON4 Schema

```
"ifcProductDefinitionShape": {
  "type": "object",
  "properties": {
    "instanceId": {
      "type": "integer"
    },
    "ifc": {
      "type": "string"
    },
    "name": {
      "oneOf": [
        {
          "type": "null"
        },
        {
          "type": "string",
          "maxLength": 255
        }
      ]
    },
    "description": {
      "type": [
        "string",
        "null"
      ]
    },
    "representations": {
      "type": "array",
      "items": {
        "type": "object",
        "allOf": [
          {
            "$ref": "#/properties/ifcShapeRepresentation"
          }
        ]
      },
      "minItems": 1
    }
  },
  "required": [
    "name",
    "description",
    "representations"
  ]
}
```

4.5.2 Product Placement Data

In IFC4 EXPRESS schema, product occurrences can be placed in 3D space relative to where they are contained. Placement is defined by a relative position (X, Y, Z coordinates), a horizontal reference direction, and a vertical axis direction [60]. The ObjectPlacement attribute that exists in IfcProduct (a high-level supertype of IfcColumn)

establishes the coordinate system in which all points and directions used by the geometric representation items under Representation are founded [60]. ObjectPlacement attribute in the IFC4 EXPRESS Schema is an optional IfcObjectPlacement that is the abstract supertype of both IfcGridPlacement and IfcLocalPlacement. In this ifcJOSN4 Schema, ObjectPlacement is narrowed down to IfcLocalPlacement.

In the IFC4 EXPRESS specification, IfcLocalPlacement is specified with two attributes as "PlacementRelTo" which is optional and "RelativePlacement" which type is an IfcAxis2Placement2D. Table 18-Right shows the ifcJSON4 schema definition for ifcLocalPlacement. Here, the IfcAxis2Placement which is a select datatype in EXPRESS, is shown by its select types (i.e. IfcAxis2Placement2D and IfcAxis2Placement3D) as discussed earlier. In addition, since the "PlacementRelTo" in IfcLocalPlacement is optional, in this use case "PlacementRelTo" is narrowed down to its null value.

Table 17 IfcLocalPlacement in an SPF file

IFC SPF format data

`#6= IFCCARTESIANPOINT((0.,0.,0.));`
`#31= IFCAXIS2PLACEMENT3D(#6,$,$);`
`#257= IFCLOCALPLACEMENT($,#31);`

The SPF representation of IFC entities in Table 17 shows the IfcLocalPlacement entities and its related information. This is a simple representation of product placement in the SPF file. Also, Table 18-Left shows the data in the ifcJSON document that represents the same IfcLocalPlacement data as in the SPF file (i.e. Table 17). As mentioned, the "PlacementRelTo" is considered as null since it is an optional attribute in the EXPRESS schema. This document passes the validation against the ifcJSON4 schema in Table 18-Right.

Table 18 IfcLocalPlacement in ifcJSON document (Left), ifcJSON4 schema representation for IfcLocalPlacement (Right)

ifcJSON document

```
{
    "ifcLocalPlacement": {
        "instanceId": 257,
        "ifc": "IFCLOCALPLACEMENT",
        "placementRelTo": null,
        "relativePlacement": {
            "instanceId": 31,
            "ifc": "IFCAXIS2PLACEMENT3D",
            "location": {
                "instanceId": 6,
                "ifc": "IFCCARTESIANPOINT",
                "coordinates": [0, 0, 0]
            },
            "axis": null,
            "refDirection": null
        }
    }
}
```

ifcJSON4 Schema

```
"ifcLocalPlacement": {
  "type": "object",
  "properties": {
    "instanceId": {
      "type": "integer"
    },
    "ifc": {
      "type": "string"
    },
    "placementRelTo": {
      "type": "null"
    },
    "relativePlacement": {
      "oneOf": [
        {
          "type": "object",
          "allOf": [
            {
              "$ref": "#/properties/ifcAxis2Placement2D"
            }
          ]
        },
        {
          "type": "object",
          "allOf": [
            {
              "$ref": "#/properties/ifcAxis2Placement3D"
            }
          ]
        }
      ]
    }
  },
  "required": [
    "placementRelTo",
    "relativePlacement"
  ]
}
```

4.5.3 Owner History Data

In the IfcRoot which is the most abstract and root class for all IFC entity definitions [60], ownership is captured in "OwnerHistory" attribute using IfcOwnerHistory. OwnerHistory assigns the information about the current ownership of that object, such as owning actor and application. The IFC4 Schema definition for IfcOwnerHistory declares attributes such as "OwningUser", "OwningApplication" illustrated in Figure 23. In ifcJSON4 Schema (Appendix 1) for IfcOwnerHistory both IfcPersonAndOrganization and IfcApplication are referenced in this part of the schema.

Table 19 IfcOwnerHistory in an IFC SPF file

IFC SPF format data
#1= IFCPERSON('ID','Afsari','Keresh',$,$,$,$,$); #2=IFCORGANIZATION('organization','Organization name', 'Organization description',$,$); #3= IFCPERSONANDORGANIZATION(#1,#2,$); #5= IFCAPPLICATION(#2,'2016','Application','Application identifier'); #41= IFCOWNERHISTORY(#3,#5,$,.NOCHANGE.,$,$,$,1407386573);

For this use case, some of the optional attributes in IfcPersonAndOrganization, IfcApplication, and IfcPerson are narrowed down to their null values. These optional attributes include: IfcOrganization.Roles, IfcOrganization.Addresses, IfcPersonAndOrganization.Roles, IfcPerson.Roles, IfcPerson.Addresses.

In an IFC SPF file, the IfcOwnerHistory can be represented similar to data in Table 19. This lists the IfcApplication, IfcPersonAndOrganization, IfcOrganization, and IfcPerson. The ifcJSON document in Table 20-Left shows the data representation for the IfcOwnerHistory in ifcJSON format which passes the validation against the ifcJSON4 schema in Table 20-Right and carries the same data as in the SPF instance file in Table 19. As discussed above, the values for some attributes such as IfcOrganization.Roles, IfcOrganization.Addresses, IfcPersonAndOrganization.Roles, IfcPerson.Roles, IfcPerson.Addresses are set to null since they are optional in IFC4 EXPRESS Schema.

Table 20 IfcOwnerHistory in ifcJSON document (Left), ifcJSON4 schema representation for IfcOwnerHistory (Right)

ifcJSON document

```
{
    "instanceId": 41,
    "ifc": "IFCOWNERHISTORY",
    "owningUser": {
        "instanceId": 3,
        "ifc": "IFCPERSONANDORGANIZATION",
        "thePerson": {
            "instanceId": 1,
            "ifc": "IFCPERSON",
            "identification": "KID",
            "familyName": "Afsari",
            "givenName": "Keresh",
            "middleNames": null,
            "prefixTitles": null,
            "suffixTitles": null,
            "roles": null,
            "addresses": null
        },
        "theOrganization": {
            "instanceId": 2,
            "ifc": "IFCORGANIZATION",
            "identification": "organization",
            "name": "Organization name",
            "description": "Org description",
            "roles": null,
            "addresses": null
        },
        "roles": null
    },
    "owningApplication": {
        "instanceId": 5,
        "ifc": "IFCAPPLICATION",
        "applicationDeveloper": {
            "instanceId": 2,
            "identification": "organization",
            "name": "Organization name",
            "description": "Org description",
            "roles": null,
            "addresses": null
        },
        "version": "2016",
        "applicationFullName": "Application",
        "applicationIdentifier": "App identifier"
    },
    "state": null,
    "changeAction": "NOCHANGE",
    "lastModifiedDate": null,
    "lastModifyingUser": null,
    "lastModifyingApplication": null,
    "creationDate": 1407386573
}
```

ifcJSON4 Schema

```
"ifcOwnerHistory": {
        "type": "object",
        "properties": {
          "instanceId": { "type": "integer" },
          "ifc": { "type": "string" },
          "owningUser": {
            "type": "object",
            "allOf": [{ "$ref": "#/properties/ifcPersonAndOrganization"
}]
          },
          "owningApplication": {
            "type": "object",
            "allOf": [{ "$ref": "#/properties/ifcApplication" }]
          },
          "state": {
            "oneOf":[
              { "type": "null" },
              { "type": "string",
                "enum": ["READWRITE", "READONLY", "LOCKED", "READWRITELOCKE
D", "READONLYLOCKED"]}
            ]
          },
          "changeAction": {
            "oneOf":[
              { "type": "null" },
              { "type": "string",
                "enum": ["NOCHANGE", "MODIFIED", "ADDED", "DELETED",
"NOTDEFINED"]}
            ]
          },
          "lastModifiedDate": {
            "oneOf":[
              { "type": "null" },
              { "type": "integer"}
            ]
          },
          "lastModifyingUser": {
            "oneOf":[
              { "type": "null" },
              {  "type": "object",
                "allOf": [{ "$ref": "#/properties/ifcPersonAndOrganizatio
n" }]
              }
            ]
          },
          "lastModifyingApplication": {
            "oneOf":[
              { "type": "null" },
              {  "type": "object",
                "allOf": [{ "$ref": "#/properties/ifcApplication" }]
              }
            ]
          },
          "creationDate": { "type": "integer"}
        },
        "required": ["owningUser", "owningApplication", "state",
"changeAction", "lastModifiedDate", "lastModifyingUser", "lastModifyingAp
plication", "creationDate"]
      }
```

4.5.4 Precast Column

The building element used in this use case is a precast column. In the IFC specification, a column is represented by either IfcColumnStandardCase or IfcColumn mostly used for columns with changing profile sizes along the extrusion [60].

Table 21 IfcColumn instance in an IFC SPF file

IFC SPF format data
#41= IFCOWNERHISTORY(#3,#5,$,.NOCHANGE.,$,$,$,1407386573); **#252= IFCPRODUCTDEFINITIONSHAPE($,'DetailedProfile',(#111));** **#257= IFCLOCALPLACEMENT($,#31);** **#259= IFCCOLUMN('3$gxit421D9RMef03b0mxA',#41,'Precast Concrete Column'** **,$,'Column',#257,#252,'267108', 'COLUMN');**

Here, to refer to the precast column, IfcColumn is used that inherits attributes from IfcRoot, IfcElement, and IfcBuildingElement. In IFC4 EXPRESS definition, IfcColumn defines “PredefinedType” attribute as an enumerator. In ifcJSON4 Schema definition (Appendix 1) illustrated in Table 22-Right, the “OwnerHistory”, “ObjectPlacement”, and “Representation” are the attributes that are referring to “ifcOwnerHistory”, “ifcLocalPlacement”, and “ifcProductDefinitionShape” respectively.

A simple representation of IFC SPF data for IfcColumn with its corresponding placement (i.e. IfcLocalPlacement), geometry representation (i.e. IfcProductDefinitionShape), and OwnerHistory (i.e. IfcOwnerHistory) is shown in Table 21. The ifcJSON document for this IfcColumn instance is shown in Table 22-Left with references to the corresponding sections for placement, geometry representation and OwnerHistory data content. This document includes the same data in SPF format shown in

Table 21 and passes the validation against the ifcJSON schema (Appendix 1) illustrated in Table 22-Right.

Table 22 IfcColumn in ifcJSON document (Left), ifcJSON4 schema representation for IfcColumn (Right)

ifcJSON document

```
{
  "instanceId": 259,
  "ifc": "IFCCOLUMN",
  "globalId": "3$gxit421D9RMef03b0mxA",
  "ownerHistory": { // see section 4.5.3},
  "name": "Precast Concrete Column",
  "description": null,
  "objectType": "Column",
  "objectPlacement": {// see section 4.5.2},
  "representation": {// see section 4.5.1},
  "tag": "267108",
  "predefinedType": "COLUMN"
}
```

ifcJSON4 Schema

```
"ifcColumn": {
    "type": "object",
    "properties": {
      "instanceId": { "type": "integer" },
      "ifc": { "type": "string" },
      "globalId": {
          "type": "string",
          "maxLength": 22
      },
      "ownerHistory": {
          "oneOf":[
            { "type": "null" },
            {  "type": "object",
              "allOf": [{ "$ref": "#/properties/ifcOwnerHistory" }]
            }
          ]
      },
      "name": {
          "oneOf":[
            { "type": "null" },
            { "type": "string",
              "maxLength": 255}
          ]
      },
      "description": {"type" : ["string", "null"]},
        "objectType": {
          "oneOf":[
            { "type": "null" },
            { "type": "string",
              "maxLength": 255}
          ]
      },
      "objectPlacement": {
          "oneOf":[
            { "type": "null" },
            {  "type": "object",
              "allOf": [{ "$ref": "#/properties/ifcLocalPlacement" }]
            }
          ]
      },
      "representation": {
        "type": "object",
        "allOf": [{ "$ref": "#/properties/ifcProductDefinitionShape" }]

      },
      "tag": {
          "type": "string",
          "maxLength": 255
      },
      "predefinedType": {
        "oneOf":[
          { "type": "null" },
          { "type": "string",
          "enum": ["COLUMN", "PILASTER", "USERDEFINED", "NOTDEFINED"]}
          ]
        }
      },
      "required": ["globalId", "ownerHistory", "name", "description",
"objectType", "objectPlacement", "representation", "tag", "predefinedType
"]
}
```

4.6 Validation for ifcJSON Schema and Document

Since JSON has been proven to be an obvious choice over XML in several studies [61, 47, 63, 64] the main objective of this chapter was to outline how IFC specification can be represented in JSON format. The data mapping section and the use case approach

explained the implementation of the IFC standard as a JSON schema to guide the creation of JSON documents so that these JSON documents can be used for web-based data transfer where JSON data needs to replace the XML documents.

It needs to be highlighted that the ifcJSON schema and ifcJSON document should be validated for JSON formatting, against JSON schema, and should be defined correctly when comparing with IFC schema based on the methodology mentioned earlier in section 4.4.3. As Table 23 indicates, ifcJSON4 schema developed in this study both in parts (i.e. the schema for three IFC concepts as geometry, placement and owner history data) and as a whole (i.e. the schema for precast column) passed the validation for formatting using a JSON data validator. In addition, ifcJSON4 schema is validated against the JSON schema. This shows that ifcJSON4 schema is a well-formed and valid JSON schema.

Table 23 Validation of ifcJSON Schema and Document

	ifcJSON4 Schema: Data Validation for Formatting	**ifcJSON4 Schema: Validation against JSON Schema**	**ifcJSON Document: Data Validation for Formatting**	**ifcJSON Document: Validation against ifcJSON4 Schema**
Validation Tool	jsonlint.com	JSON Schema Lint	jsonlint.com	ajv
Geometry Representation	✓	✓	✓	✓
Product Placement	✓	✓	✓	✓
Owner History	✓	✓	✓	✓
Precast Column	✓	✓	✓	✓

On the other hand, in the use case approach, four sets of JSON data have been developed to represent geometry, placement, owner history and the building element (i.e. precast concrete column). This ifcJSON data as described before needs to be validated both

for JSON formatting and against ifcJSON4 schema. Table 23 lists the validation of these four sets of ifcJSON documents that were validated for formatting and against the developed ifcJSON4 schema. The result indicates that ifcJSON documents are valid JSON documents and can be used for web-based data transfer. This validation shows the result of ifcJSON implementation is a complete JSON Schema and document. If the data is provided wrong or incomplete in the ifcJSON document, the validation will report a failure. For instance, in defining "ObjectPlacement" attribute for IfcColumn as shown in Table 24-Left, if "RefDirection" attribute is not provided, the report will show an error illustrated in Figure 29. The "RefDirection" attribute is optional and it should be provided either as null or with specific value but in Table 24-Left "RefDirection" is missing. Therefore, the validation report (Figure 29) shows that ObjectPlacement has errors because based on ifcJSON4 schema, shown in Table 24-Right ObjectPlacement should be either null or match "ifcLocalPlacement" specification and ifcLocalPlacement should be either an IfcAxis2Placement2D or an IfcAxis2Placement3D entity. However, since the definition in Table 24-Left is missing the "RefDirection" property, it does not match any of the specified options in ifcJSON4 schema specification.

Table 24 Instance definition for ifcColumn.ObjectPlacement (Left), ifcJSON4 schema representation for ifcColumn.ObjectPlacement (Right)

ifcJSON document

```
"objectPlacement": {
    "instanceId": 257,
    "placementRelTo": null,
    "relativePlacement": {
        "instanceId": 31,
        "location": {
            "instanceId": 6,
            "coordinates": [0,0,0]
        },
        "axis": null
    }
},
```

ifcJSON4 Schema

```
"objectPlacement": {
    "oneOf":[
        { "type": "null" },
        {  "type": "object",
           "allOf": [{ "$ref": "#/properties/ifcLocalPlacement" }]
        }
    ]
},
```

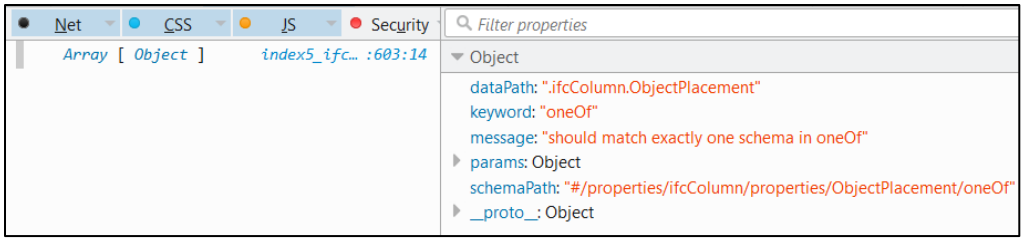


Figure 29 Validation report for ifcColumn data represented in Table 24-Left

Similarly, Table 25-left shows parts of an ifcJSON document that does not pass the validation against ifcJSON4 schema. In this example, “Representations.ContextOfItems” attribute is set to null. As described in Figure 24, Body SweptSolid Geometry concept definition in IFC4 specification requires IfcGeometricRepresentationContext to be provided for geometric representation of SweptSolid and this specification requirement is implemented in ifcJSON4 schema illustrated in Table 25-Right. Therefore, the validation of data in Table 25-Left against ifcJSON4, will report an error on “ContextOfItems” as shown in Figure 30. The missing properties will be listed in the validation report one after each other until all the required properties are provided. In this report (Figure 30) “ContextIdentifier” is shown as a missing property. When “ContextIdentifier” is added, the next missing property (i.e. "ContextType") will be reported until all properties are complete.

These examples describe how ifcJSON4 schema can be used to generate and validate ifcJSON documents.

Table 25 Left: Instance definition for ifcColumn.Representation.Representations.ContextOfItems, Right: ifcJSON4 schema representation for ifcColumn.Representation.Representations.ContextOfItems

ifcJSON document	ifcJSON4 Schema

```
"representations":[{
        "instanceId": 111,
        "contextOfItems": {},
        "representationIdentifier": "Body",
        "representationType": "SweptSolid",
        "items": [{
            "instanceId": 110,
            "sweptArea": {
                "instanceId": 108,
                "profileType": "AREA",
```

```
"ifcShapeRepresentation": {
    "type": "object",
    "properties": {
        "instanceId": { "type": "integer" },
        "contextOfItems": {
            "type": "object",
            "allOf": [{ "$ref": "#/properties/ifcGeometricRepresentationContext" }]
        },
        "representationIdentifier": {"type" : ["string", "null"]},
        "representationType": {"type" : ["string", "null"]},
        "items": {
            "type": "array",
            "items": {
                "type": "object",
                "allOf": [{ "$ref": "#/properties/ifcExtrudedAreaSolid" }]
            },
            "minItems": 1
        }
    },
    "required": ["contextOfItems", "representationIdentifier", "representationType", "items"]
},
```

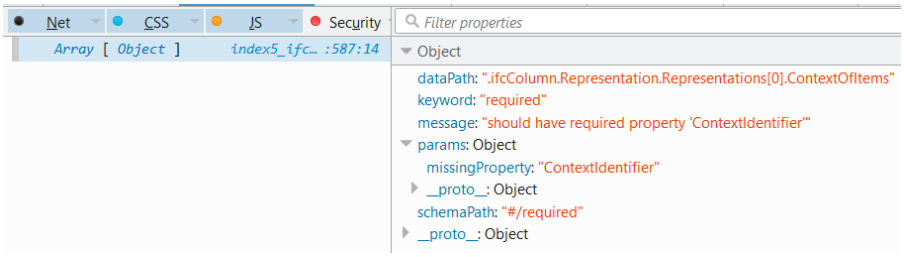


Figure 30 Validation report for ifcColumn data represented in Table 25-Left

CHAPTER 5. RESTFUL IFC API

This Chapter introduces a new approach to BIM data exchange in Cloud that utilizes web technologies to access and transfer model-based BIM data. Rather than exposing BIM data through proprietary and tightly-coupled systems in Cloud, this research proposes to make them an integral part of the Web. To achieve that, this study suggests to expose the resources made available by Cloud-BIM platforms through a RESTful API and represent resources based on IFC specification as the open BIM standard.

5.1 The Need for a RESTful API

Application Programming Interfaces (APIs) can provide access to the capabilities of web services where they are exposed and if Cloud applications share a common set of APIs they will be able to interoperate [37, 16]. Cloud APIs specify application-to-application interaction so that other applications can request data from the platforms [17]. Therefore, communication among programs is enabled through Cloud APIs [82]. However, to ensure Cloud interoperability, two major aspects need to be considered when dealing with Cloud APIs: first, the APIs need to be designed and accessed with minimizing coupling in mind [83] because conventional approaches that build tightly coupled systems have faced many challenges [84]. Second, each Cloud provider has its own Cloud API which causes inconsistencies so, Cloud services should use a common standard to harmonize their APIs [16, 17].

To address the first aspect, this research proposes to implement a web API based on Representational State Transfer (REST) principles in the form of web services for

Cloud-BIM applications. As the web paradigm moves from Web 2.0, social networking web, to Web 3.0, ubiquitous computing web, it becomes more apparent that the need for data-on-demand considerably increases so that it could address sophisticated queries [85]. Providing loose coupling is the main goal of ubiquitous computing to support interaction with resources that are made accessible through the Internet of Things (IoT). This Provides data transfer capabilities that is required for network-based information systems [84]. The focus of IoT has been mainly on setting up connectivity in diverse and constrained networking environments. IoT then requires focusing on the application layer to build on top of network connectivity [86]. To achieve an application layer for IoT, one approach known as Web of Things (WoT), is to provide resources through standard web-based mechanisms to make the resources accessible and integrate the applications [84] and loosely coupled API in the form of web service can provide access to the resources. But there are several ways to design web systems and Cloud APIs can expose their interfaces using a variety of web technologies such as Representational State Transfer (REST), Simple Object Access Protocol (SOAP) [17]. REST is an architectural style for network-based applications [87] that is based on stateless client-server communication and specifies how resources should be defined and accessed through a network. REST has major advantages in comparison with state-based applications [84]. REST avoids application state, ensures that application resources are represented and identified by Uniform Resource Identifiers (URI), and sustain all state information that is required within the interactions between client and server [84, 88]. REST uses URIs to encapsulate resources and identify services and uses HTTP as the application protocol thus, it is a good choice

for building APIs for WoT [86]. Also, coupling between client and server are minimized in RESTful APIs [83].

To address the second aspect, as mentioned, Cloud services should use a common standard to harmonize their APIs [16, 17]. In the architecture, engineering and construction (AEC) industry, the Industry Foundation Classes (IFC) specification is the industry open standard and is now accepted as ISO 16739 standard [21]. IFC is supposed to facilitate cross-platform BIM interoperability. Therefore, the main goal in this study is to use IFC as the common standard and to focus on creating a REST API based on IFC schema to address harmonization of the Cloud-BIM APIs. This API can enable model-based BIM data to be accessed by the receiving client through web technologies.

Therefore, this chapter outlines how to design a RESTful API for enabling the exchange of web-based BIM data conforming to IFC specification. This research does not make the assumption that Cloud-BIM applications must offer RESTful interfaces directly, rather it proposes to expose the resources made available by Cloud-BIM platforms through a RESTful API to support interoperability.

5.2 REST Architectural Style

REST, as introduced initially in Roy Fielding's PhD dissertation [87], is an architectural style for network-based applications and is not a specific set of technologies [86, 88]. REST follows the architectural principles of the web and is known as conceptualization of the web as a loosely coupled system [88]. REST supports creating loosely coupled web systems that can be reused [86]. Web API based on REST architecture specifies how the outside world can remotely use the application's functionality and

resources and in fact, RESTful API provides a type of access to the data that is more lightweight [89]. RESTful web services have several advantages such as simplicity, loose coupling, interoperability, scalability and serendipitous reuse [88]. RESTful interfaces are provided in several web services and Cloud applications and their back-ends can follow different models and database systems. WoT can also implement the same approach [86] and in fact, RESTful architecture is very effective for the WoT [89].

Resources are the application components that should be used or addressed and are the fundamental concepts in REST [89, 86]. REST manages all interactions between client and server as exchanges of resource representations [84]. Resources can include physical objects, abstract concepts, or dynamic concepts such as server-side state [89, 86].

In the design of a RESTful system a set of constraints which are defined in REST architectural style should be applied [88]. Major constraints are as follows:

- Resource Identification: As Fielding explains, any information such as a document or an image that can be named can be a resource. Some resources are static and always correspond to the same value set while others might have values that dynamically change over time. The semantics of mapping for a resource must be static because the semantics identify each resource and distinguish resources from each other [87]. All resources of an application should have a unique and stable identifier. To establish an identification scheme, REST uses URIs [89, 86] which should be global identifiers to be referenced and dereferenced independent of context [88].

- Uniform Interface: A uniform interface with HTTP interaction semantics provides access to resources [89, 86]. Unlike other types of APIs that use API methods, REST uses general set of HTTP methods, so, interactions with resources can only be through the uniform interface or a subset of that [88]. HTTP has limited methods as GET, PUT, POST, and DELETE that effectively optimizes the interactions [89, 86].
- Self-describing Messages: Each client request and server response is a message that should be self-descriptive and each message should have a body and metadata [88]. Message types should also be understood by both client and server [89]. HTTP outlines message types and metadata elements in HTTP headers.
- Resource Representation: resource representation should include the important aspects of a resource to provide complete understanding of the resource through its representation [88]. Resource representation formats such as XML or JSON should be agreed to facilitate interactions [89, 86].
- Hypermedia Driving Application State: Hypermedia refers to links to other data. In REST, resource representations should be linked in a way that the client of RESTful service can find the links and use them for interaction with other resources [88, 89]. This connectedness should be supported by resource representation within well-defined ways to expose the links [86].
- Stateless Interactions: All client/server interactions must be self-contained [88] and all information needed to execute the request must be included in the request itself. This constraint is covered in using HTTP since it is based on request and response interaction [89] with no HTTP sessions or transactions [86].

Hypermedia As The Engine Of Application State (HATEOAS) or the hypermedia constraint is the most important constraint in REST architecture style that supports loose coupling. Based on this constraint, clients can interact with resources dynamically and there is no need for previous agreements [88].

5.3 Cloud-BIM and RESTful API

A RESTful API is a lighter, faster and more efficient API compare to other alternatives. Its focus is on accessing identified resources through a uniform interface which does not need expensive tools to interact with the Cloud application. Any application can access available resources of the RESTful API in real-time. In the AEC industry, the importance of Cloud APIs has become more evident with the growing adoption of BIM and Cloud-based services. Standard RESTful APIs can expose BIM data to enable users to access available data fast and in real-time. REST APIs separate the client and its user interface from the server and from the data storage so it increases scalability of the BIM projects and allows server and client BIM applications to apply changes on their components independently. There are several vendors that offer BIM applications and recently developed Cloud-based services. The separation between client and server in RESTful APIs makes the development of individual BIM applications possible with no dependencies on each other while the REST API provides interoperability among these applications. In this loosely coupled collaboration, any application can migrate to other server or make changes in their components or their database and the interoperability can remain intact if the Cloud-BIM application provides the data for each API request. This makes the BIM development more flexible. Moreover, the users can deploy different Cloud-BIM services from different vendors more easily when they know that these Cloud-

BIM applications provide a REST API with identified resources that can be easily exchanged. So, the collaboration among project parties in design and construction process can be facilitated.

Besides, REST API is independent of the type of Cloud platform or languages. It can adapt to the syntax and platforms of any collaborating BIM applications so it can be used and integrated with any existing or future Cloud-BIM applications. The only thing is that to support interoperability, the same resource representation and resource identification for the REST API should be used in collaborating Cloud-BIM applications.

5.4 Examples of Similar REST APIs

There are many popular services that offer RESTful APIs such as Basecamp, Flickr, and Google Maps. Basecamp is a web-based application for project management, mainly dealing with people, shared documents, activities and time. Basecamp has a REST-style API and every resource in the API such as a project's update or project's to-do list has a URI [90]. For its resource representation, it uses a proprietary XML format in its classic API but Basecamp 2 and 3 API uses JSON serialization. Its workflow is not built around an individual industry and it can be used for any industry. There is similarity of representation structure that is similar between Basecamp project management and AEC project management tools such as the workflow, roles and deliverables.

Flickr is a website for hosting image and video which deals with shared media content, people, groups, and tags. It has a REST API that provides its main resources such as photos using URIs. It offers proprietary XML and JSON as its resource representation [90]. In the building construction process, images and videos has previously served as a

visual reference to site progress. Recently, photogrammetry and creating 3D geometry from images have become possible so images from a construction site can be processed to create 3D data. Drones are also offering ways to collect aerial images and therefore, such data is becoming integrated with BIM tools to verify the construction process against the BIM model. Flickr's REST API and resource representation can be considered as an example for the construction domain when integrating image and video data with the BIM process in Cloud-based applications.

The Google Maps API which consists of a group of APIs deals with geographic data for maps applications. In its RESTful API, each resource such as address and geo-location is exposed through the URIs with JSON and proprietary XML representation [90]. Google converts addresses into geographic coordinates. Also, its data layer provides a container for arbitrary geospatial data and styling information. Google Maps data contains geometries, such as points, lines or polygons and the user can also style these geometry features and add other properties. This has similarities to the BIM model in terms of the uses of geometry data. Therefore, the design and usage of Google Maps API, its resource representation and URI pattern is a directly useful example for the AEC industry and Cloud-BIM applications.

3DRepo is another example which is an open source Version Control System (VCS) for 3D assets that is offered through a RESTful service [91]. It is provided for the construction industry in the UK mainly to manage digital 3D data for cost reduction and carbon emission [92]. 3DRepo framework is introduced in collaboration with Arup and focuses on unifying storage and control thus uses a NoSQL database to store 3D assets and their revision histories [93]. It converts a 3D file to scene graph components and then for

storing and revision tracking it encodes the data to a Binary JSON (BJSON) format [94] with a proprietary schema. It is a file server that works on exporting and importing 3D models and is not aimed at addressing the interoperability issue in Cloud services. But its REST API has demonstrated that clients can connect to a portal to access version controlled 3D data from different sources and its RESTful service with resource-based approach can support delivery of requested 3D assets [93].

These example APIs indicate that the implementation of REST architectural style for Cloud-BIM APIs can deliver AEC required data such as geometry, scheduling and other similar entities. In addition, to ensure that the AEC semantics are included in the REST API, the design and semantics of the API's resources for Cloud-based BIM applications should follow an industry approved open standard i.e. IFC specification.

5.5 The Design for a RESTful IFC API

The most important parts of the web's architecture are URIs for resource identification as well as the HTTP as the main protocol for interacting with resources in a lightweight and loosely coupled way [84]. In REST architecture, resources are the core and everything is modelled as resources while each resource is identified with a unique URI. A resource is an object with relationships to other resources. The difference between a REST resource and an object in an object-oriented programming language is that REST resources only work with a few methods (i.e. HTTP calls). In Fielding's REST style, a resource is "a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time" [87]. In designing a RESTful API three important aspects should be considered: a) resource representation; b) the identification of resources

and; c) naming scheme for the resources which should be made available to specify the URI patterns. Thus, defining REST resources is a core part of making things available on the web [84]. Also, as mentioned earlier, for interaction with other resources, resource links to other resources should be defined [88, 89]. This section describes the design for IFC RESTful API regarding these four aspects: resource representation, resource identification, URI patterns, and resource links. At the end, the study will also discuss how the HTTP interactions in the API's uniform interface address BIM and AEC semantics.

5.5.1 Resource Representation

Data interchange formats in REST APIs are typically either XML or JSON and the API can have supports for both formats. As explained in the previous chapter (i.e. Chapter 4), the advantages of JSON encoding over XML should be considered. In addition, ifcJSON introduced earlier is constrained by both JSON schema as well as IFC specification as the industry-wide open standard. Therefore, this study uses ifcJSON, a JSON representation of IFC schema, for resource representation. Resources are modeled as JSON objects and data within them are modeled as key-value pair based on ifcJSON schema. The values in key-value pairs can be any of the data types that are native to JSON such as string, number, boolean, null, or arrays, explained in the previous chapter. In the REST IFC API, collections of resources are modeled as arrays of ifcJSON objects.

5.5.2 Resource Identification

In a REST API, the resource is the fundamental concept and resources can be grouped into collections. Every collection follows a single schema and it can only contain one type of resource which makes it homogenous [95]. Resources and collections have the

same semantics and in fact, collections are resources themselves. As shown in Figure 31, collections can exist at the top level of the API, or they can be sub-collections inside a single resource. Resources can be singleton resources while existing outside any collection [95].

Figure 31 Resources and collections in REST API

To identify resources, the first step is to analyze the domain and find resources that are required and will be used in the REST API [96]. This study looks at IFC specification as the industry open BIM standard that captures domain resources. The API resources should be designed based on the need of the API users and API interactions because after resource identification, API will be modeled for HTTP interactions with the identified resources [96].

5.5.2.1 IFC Concepts

IFC schema is the open BIM standard for the AEC industry. Besides, the US National BIM Standard [29] proposes facilitating information exchanges through Model View Definitions (MVDs). A Model View Definition that is specific to an IFC release, defines a subset of the IFC schema that is needed to satisfy one or many exchange

requirements of the AEC industry [97]. In fact, MVD consists of one or multiple information exchange and each information exchange is the data that must be provided by one application to support work in one or more other applications. MVD serves as the technical exchange specification and its functional requirements is captured within an Information Delivery Manual (IDM). IDM identifies and documents user needs and specification of the BIM exchanges specifying the exchange process, workflow, data sharing, data format and content requirements for one or more BIM Exchange Models [26].

In addition, concept implementation has been introduced to modularize MVD development and improve its re-usability. A subset of a schema forms a concept as a modular sub-unit that can be used to create several MVDs [27]. In fact, contents in different exchanges but within similar domains are often replicated. This creates the notion of data exchange modules that could be reused. The reusable modules represent semantic units that map the exchange pieces to an information model schema that is most often an IFC sub-schema [25]. The reusable modules known as MVD concepts or IFC concepts, are aggregated into Exchange Models that are subsets of MVD concepts specifying the information for a specific workflow exchange. Figure 32 diagrams the concept structure. These concepts are the proposed basis for the Exchange Models representing the semantic and functional knowledge of an industry domain [25] [20].

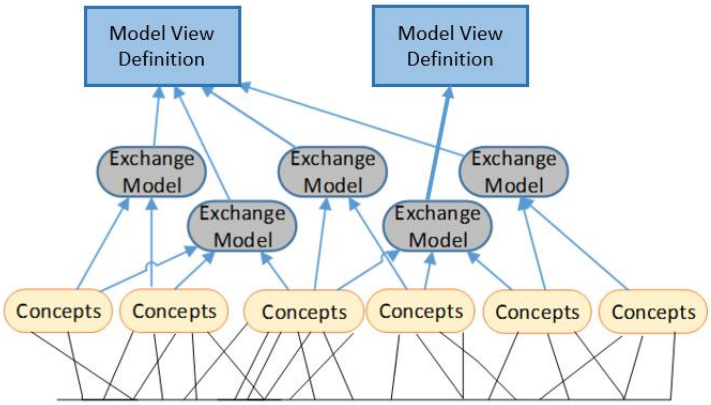


Figure 32 MVD concepts structure

In IFC4 specification which is the current version of IFC schema, common concepts are introduced to specify how data types and IFC entities can be used in different scenarios. These common concepts are applied to entities that have specific use and are defined in a graph of entities and attributes specifying the constraints and parameters required for attributes and instance types [60]. IFC4 concepts are categorized into 10 groups: project, object definition, object association, product shape, product type shape, composition, assignments, connectivity, root tracking, and resource. For instance, "Product Placement" is an IFC4 concept under the "Product Shape category" [60]. As shown in Figure 33, Product Placement concept in IFC 4 defines the entities that are needed to define how a product being placed in 3D space relative to the product's context. The product's placement as "ObjectPlacement" can be specified either relative to a grid or relative to another product's local placement.

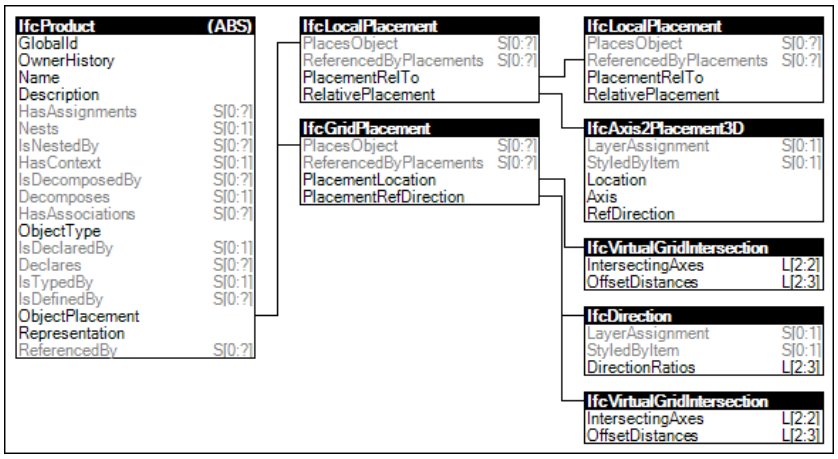


Figure 33 Product Placement concept definition in IFC4

Another example of IFC4 concepts is the "Body SweptSolid Geometry" that is defined under the "Product Shape"/ "Product Geometric Representation"/ "Body Geometry" [60]. This concept as shown in Figure 34, is a product's 3D geometric representation based on swept solid modeling and it allows for either basic extruded area solids or revolved area solids.

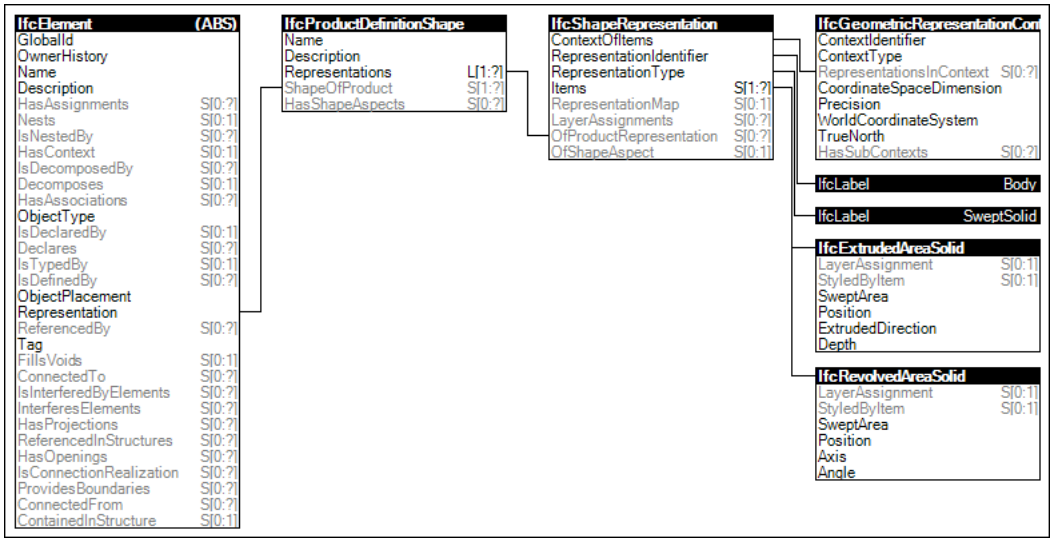


Figure 34 Body SweptSolid Geometry concept definition in IFC4

The IFC4 concepts are defined for generic use while an MVD is designed for a target domain in the AEC industry and therefore, the MVD has additional specifications that modify the scope and rules of the IFC schema. In the MVD for precast concrete domain, 93 reusable concepts are identified [77]. An example of a precast concept is shown in Figure 36 as "PCI-145" that specifies the concept for "Precast Projection Attributes". This concept defines an addition to a precast piece in the form of any projection from the normal bounds of the piece such as a corbel shown in Figure 35. The concept graph in Figure 36 specifies that the projection element must be defined with "IfcProjectionElement" entity and must satisfy the data for owner history, object type, object placement, geometric representation which is defined either in swept solid or boundary representation (B-rep) or mapped representation, as well as a piece tag.

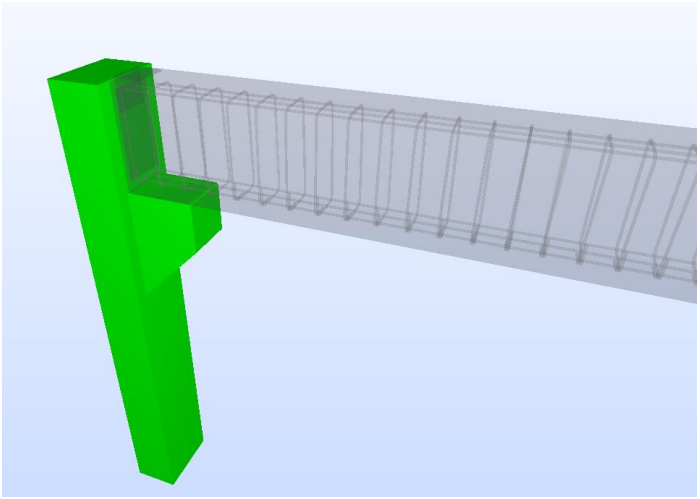


Figure 35 Precast corbel as feature element addition

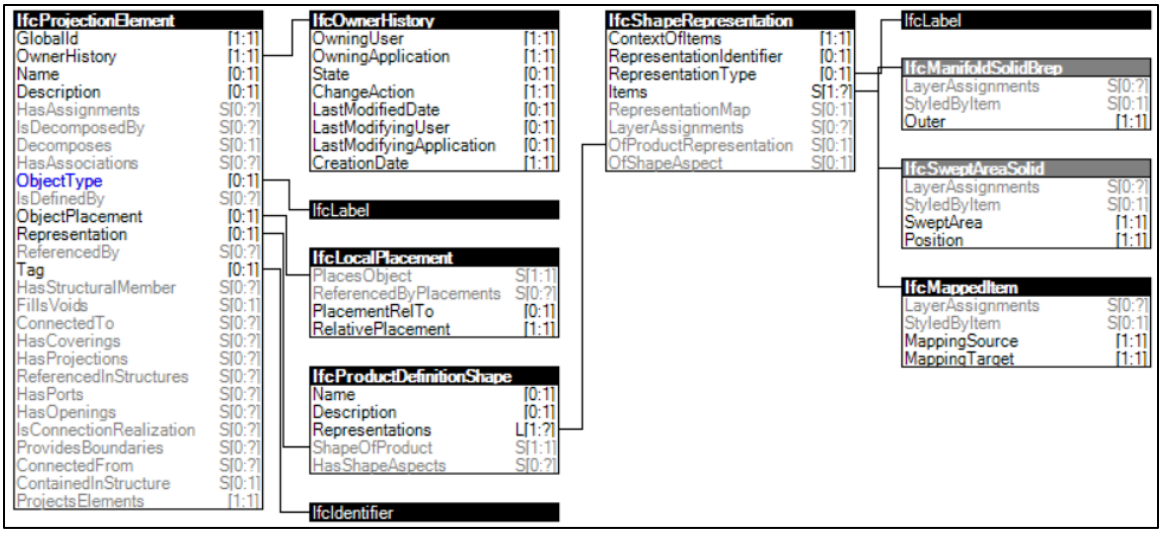


Figure 36 Concept definition for Precast Projection Attribute in precast MVD

5.5.2.2 IFC-based REST Resources

In the AEC industry, building data such as objects and processes are described in IFC data model to support interoperability. IFC specification has been the only open standard for BIM. This research is aimed at achieving a standardized solution for Cloud-BIM data exchange. Thus, since IFC is the industry-wide standard, here the objective is to design REST resources based on IFC specification. Instead of randomly designing REST resources, IFC concepts can be used to specify the schema for collections in REST resource identification. IFC concepts as the underlying basis of REST resource identification would benefit the API design because first, IFC concepts are common modules of data exchange which are standardized in the AEC industry, so this will allow a common understanding of units of data within REST resources. Second, IFC concepts provide the ability to reuse definitions and are independent of MVDs [98] while they are reusable across several exchanges; so, the use of IFC concepts for REST resource identification will ensure that REST resources can be reused. Third, all details of the BIM data can be captured in the

IFC binding of individual concepts [98] so REST resources based on IFC concepts can cover all required details of BIM data.

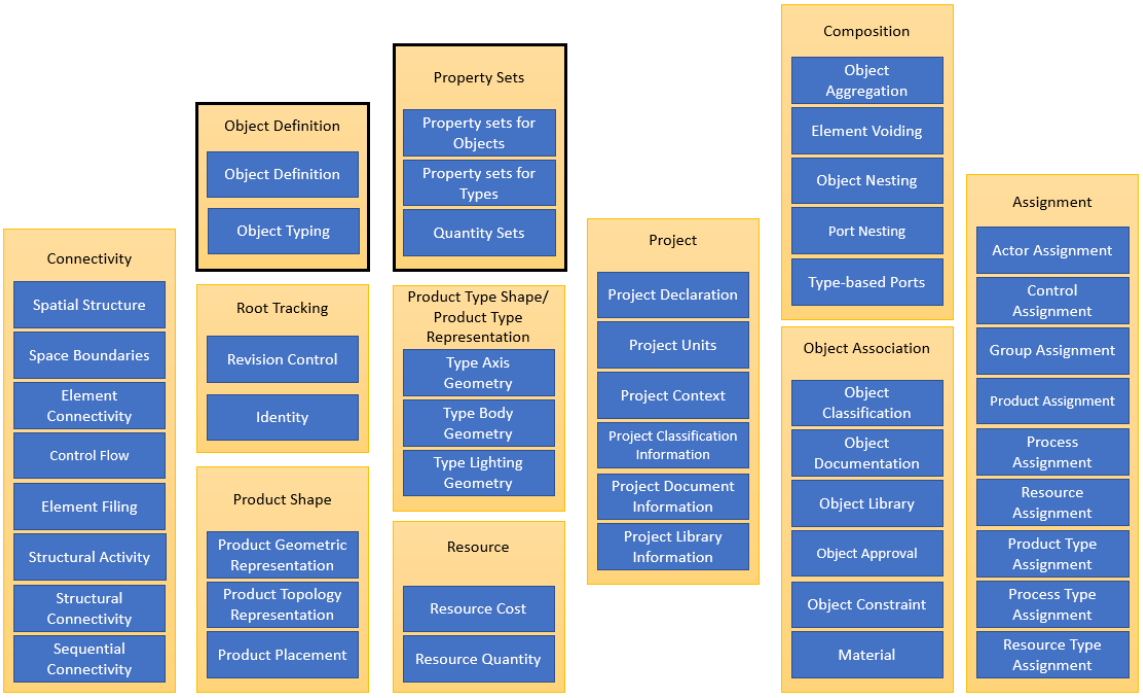


Figure 37 REST Resource identification based on IFC fundamental concepts

REST resource identification based on IFC concepts is illustrated in Figure 37. Although IFC4 Concepts are categorized into 10 groups, the REST resources are grouped into 11 main collections. The reason is that in IFC4 specification, “Property Sets” concept is defined under Object Definition but “Property Sets” collection needs to be defined separately because: 1) property set definition in projects can become very complex and also property sets have the capability for being user defined, so it will be more effective if “Property Sets” collections are defined separately 2) In IFC specification IfcRoot consists of three subtypes as IfcObjectDefinition, IfcRelationship, and IfcPropertyDefinition thus, to align REST collections with this underlying nature, “Property Sets” collection will be

kept separate. Therefore, RESTful IFC API consists of 11 main collections (Figure 37). Each main collection includes other sub-collections following IFC4 fundamental concepts.

In IFC4 specification the “Object Definition” concept is specified to define an object occurrence. But despite the presence of “Object Definition” concept in IfcDoc baseline - which is the tool generates IFC documentation- the IFC4 specification does not include any concept definition or diagrams for “Object Definition”. Since object occurrence should be defined by its own and not just by the property set and object type, this study includes the “Object Definition” as an IFC fundamental concept.

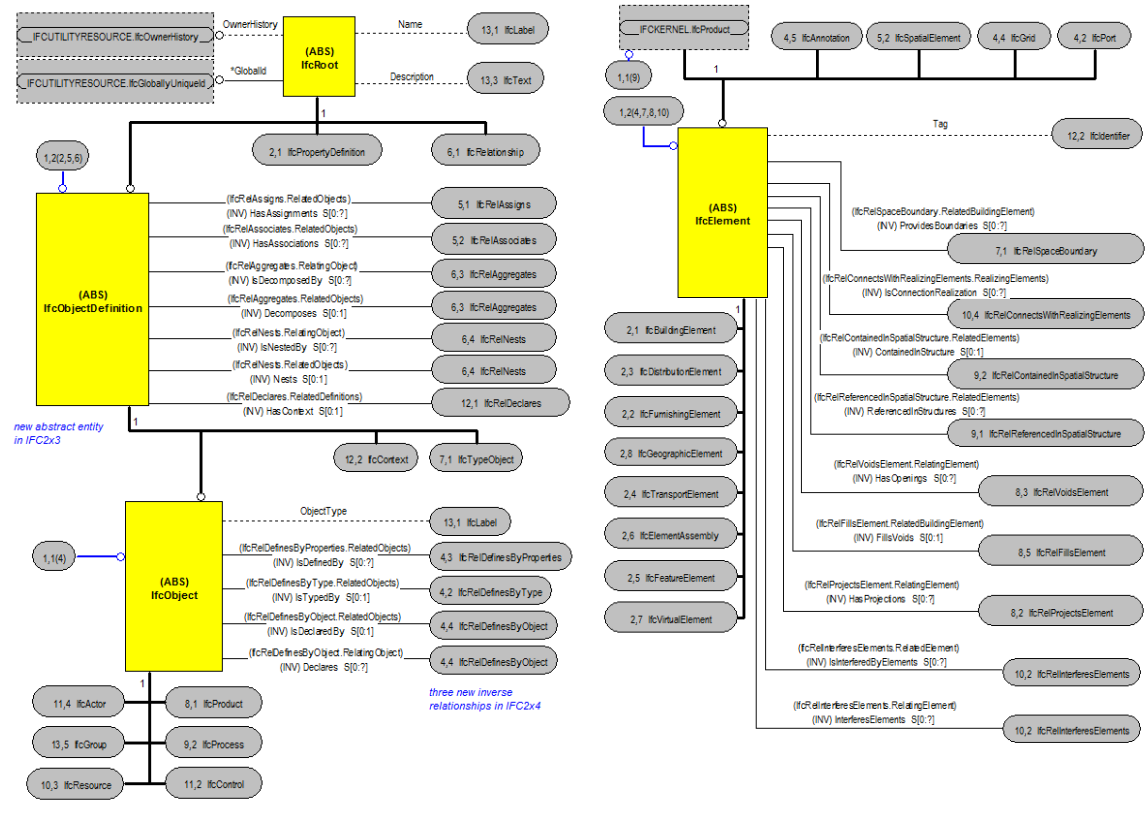


Figure 38 EXPRESS-G specification for IfcObject (left) and IfcElement (right) in IFC schema

The notion of IFC abstract entities can be captured within collections and sub-collections in REST resources. For example, IFC "Object Defintion" concept defines the object occurance and it applies on "IfcObject" which is an abstract supertype for IfcActor, IfcControl, IfcGroup, IfcProcess, IfcProduct, and IfcResource as shown in Figure 38.

Therefore, as shown in Figure 39, REST resource identification for IfcBuildingElement which is a subtype of IfcElement (shown in Figure 38 left) is specified as a sub-collection of its supertype class as IfcProduct/IfcElement/IfcBuildingElement. This resource identification strategy directly follows IFC specification.

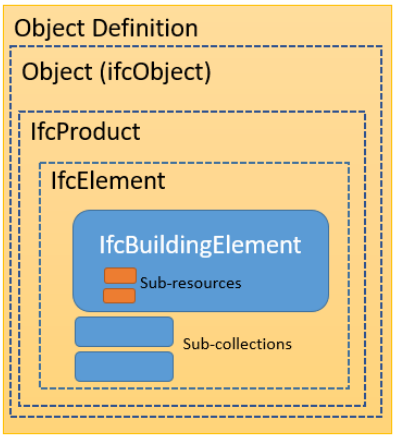


Figure 39 REST sub-collection identification based on IFC specification

When an MVD for a specific domain is used, more concepts are added to the base IFC4 fundamental concepts which adds more definition and constraints to address more specific data exchange requirements. These specific definitions and constraints can be added to the identification of REST resources. But first, the MVD concept definition should be aligned with concept definition in IFC4 specification. In the example of "Projection Element" concept (Figure 40), the concept has some overlaps with the base concepts in IFC4 such as

with “Revision Control”, “Product Geometric Representation”, and “Product Placement”. After specifying a well-defined MVD concept as “Projection Element”, the corresponding concept can be applied to the IfcProjectionElement collection within a separate collection. This can be defined under “Projection Element” collection. In fact, in the “Projection Element” concept in precast MVD, the “ObjectType” and “Tag” attribute must be provided and this constraint can be applied directly to its corresponding REST collection of resources.

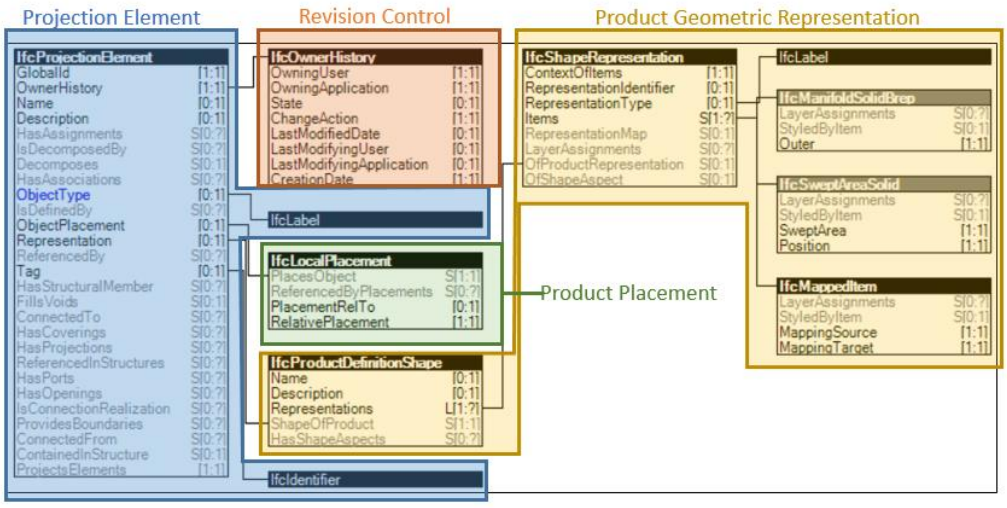


Figure 40 Precast concrete MVD concept definition for Projection Element mapped to IFC4 concepts

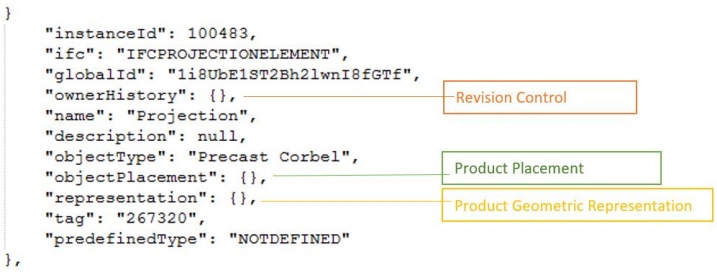


Figure 41 REST resource representation for a “Project Element” resource

An example of an “IfcProjectionElement” resource is shown in Figure 41 along with the concept definition in IFC4 in Figure 40. In this example resource, all constraints are applied based on IFC concept definition. Also in this resource, “ownerHistory”, “objectPlacement”, and “representation” properties refer to separate collections (i.e. separate IFC concept definition) “Root Tracking”/ “Revision Control”, “Product Shape”/ “Product Placement”, and “Product Shape”/ “Product Geometric Representation” collections respectively. Therefore, these attributes are illustrated as links to corresponding resources. The links structure will be discussed later in the resource links section.

Another important point to be considered is the difference between the notion of IFC concepts (or IFC concept graph) and data occurrence for REST resource identification. In the definition of IFC concepts duplicate entities are used although when using multiple concepts together, the instance occurs only once. For example, in “Product Placement” concept shown in Figure 42, IfcProduct which belongs to “Object Definition” concept seems to be duplicated. But the reason IfcProduct is shown in “Product Placement” concept is to indicate how a set of entities that define the placement of the product associates with IfcProduct itself. REST resource identification captures the notion of object occurrence directly. For instance, Figure 43 shows the resource schema for “Product Placement” and “Object Definition” collections. In “Product Placement” collection, there is no instance of IfcProduct because it belongs to “Object Definition” collection. But there needs to be explicit links to relate two REST collections, namely, “Product Placement” and “Object Definition”. The design of resource links is discussed in the following section.

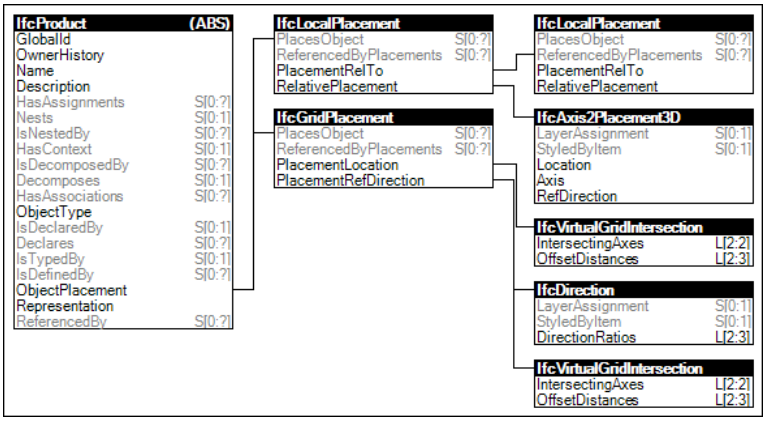


Figure 42 Concept definition for Product Placement in IFC4

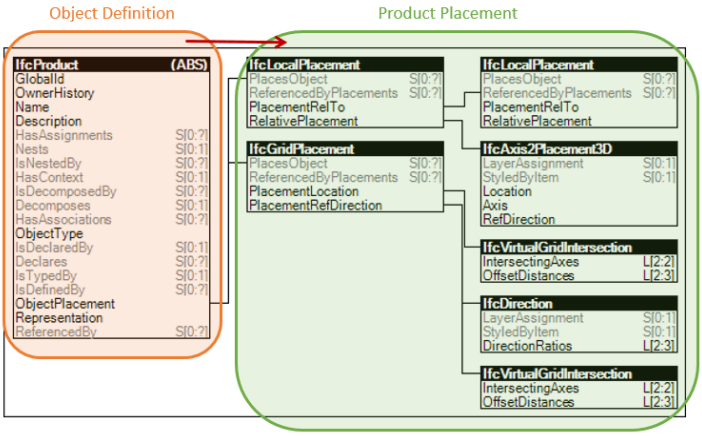


Figure 43 REST resource identification for Product Placement and Object Definition collections

Similarly, in IFC4 specification, the definition for “Body Swept Solid” concept which is specified under “Product Shape”/ “Product Geometric Representation”/ ”Body Geometry” concept mixes multiple concepts (i.e. Object Definition, Product Geometric Representation, Body Geometry, and Body SweptSolid Geometry) to capture the relationships (shown in Figure 44).

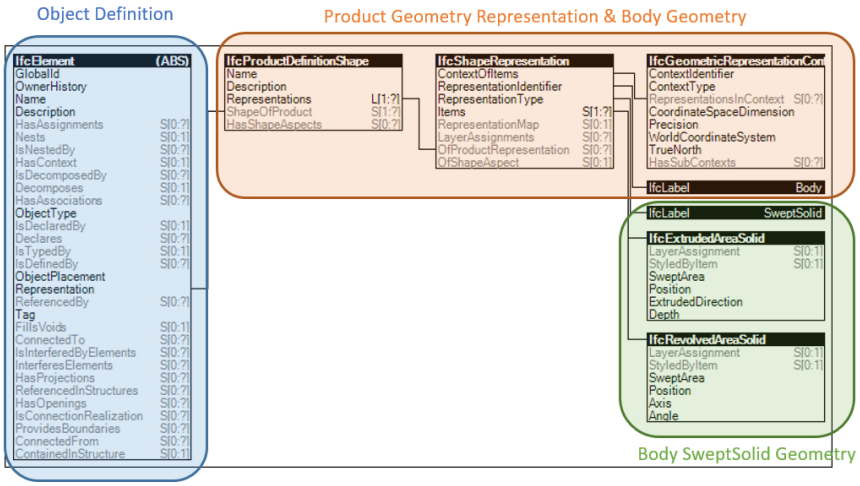


Figure 44 Body SweptSolid Geometry concept diagram in IFC specification mixes three concepts

Therefore, the distinction between underlying concepts that IFC specification vaguely combines should be noted in REST resource identification because in REST, resources do not overlap in terms of data occurrence while they are tied through links. This feature of REST design, can improve IFC rule checking capability. In IFC rule checking, since the IFC concepts have overlaps in using similar entities, these duplicated entities will be validated multiple times within the concepts that utilize the overlapping entities which make the validation process inefficient. Keeping the concepts definite and eliminating overlaps can be achieved through using REST API resources. It should be noted that IFC rule checking is not in the scope of this study and therefore, analysis of the potential improvements for BIM rule checking that can be achieved through RESTful IFC API is a future study.

5.5.3 URI Patterns

After identification of resources, the collection of resources should be named and resources naming is a critical task as the naming defines the URI patterns in the API. To access the API, the base Uniform Resource Locator (URL) should be used and the base URL for all API calls are “http://{application}/ifcrestapi/” where the application URL e.g. “www.bimapplication.com” is replaced with the {application} placeholder. Calling the base URL of the API without any options loads the main page of the API to interact with which could return a list of available resources and MVDs. Calling an MVD, will return a set of resources available based on that specific MVD. In addition, users are provided access to the API resources through URIs and URIs identify REST resources; for instance, to access building elements in a BIM model, if these entities are modelled under a collection of resources with “buildingelements” URI, the user can add this URI to the API URL in the address bar of a web browser, as shown below, to get access to the collection data.

```
/ifcrestapi/buildingelements
```

Similarly, to access the data related to a single building element in a BIM model the ID of that specific building element should be added as shown below.

```
/ifcrestapi/buildingelements/{buildingelements_id}
```

Most of the API calls are simply URLs, which can be entered in the address bar of a web browser to perform. Basically, everything used after the “/ifcrestapi/” is the URI for a collection of resources (or a specific resource) in the REST API.

A URI is an identifier that consists of a sequence of characters. Here, instead of randomly named URIs and documentation of thousands of API resources, this study

emphasizes that the URIs must be harmonized among APIs based on industry-wide standards to allow a common understanding of the available BIM resources on web. Therefore, the URI pattern can follow the naming patterns in IFC concept definition that also corresponds to REST resource identification. For example, in RESTful IFC API, the URI for resources that define building elements follows the structure and naming of its corresponding IFC concepts that was shown previously in Figure 39. This URI is as follows

```
/ifcrestapi/objectdefinitions/ifcproducts/ifcelements/ifcbuildingelements
```

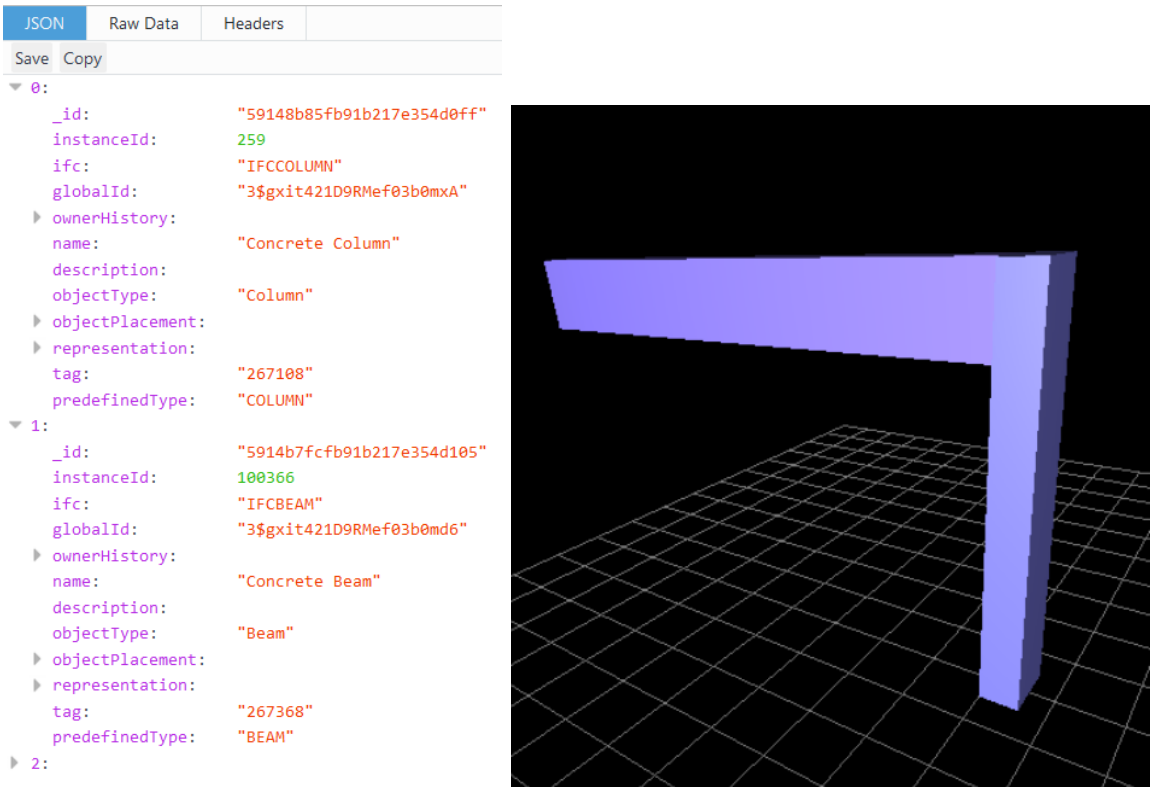


Figure 45 Left: Example of the API response for retrieving ifcbuildingelements resources, Right: Corresponding building elements

Using this URI can return all the ifcbuildingelements resources in the collection. An example of the data response from the REST API is shown in Figure 45. Every resource has its own ID listed under the "_id" key and the "objectType" property defines what type of building elements (e.g. columns, beams, etc.) the ifcbuildingelements is specified.

Also, to access resources that define the data regarding local placement for a specific piece, the URI will follow the corresponding IFC concepts as shown below.

```
/ifcrestapi/productshapes/productplacements/{productplacements_id}
```

Also, for the URI pattern, the REST API should follow common URI conventions for instance a URI cannot contain a literal space; In a URI, uppercase letters are equivalent to lowercase; And collections are named as plural entities such as "elements" rather than "element".

Here, the REST API design uses the IFC specification to identify resources, then applies the resource identification to lead the URIs patterns that follow the patterns in IFC concepts with common adjustments based on URI convention. Next, HTTP actions like GET are applied to retrieve data. This design makes the API clear and easy to discover while IFC concepts perform as interaction patterns with RESTful IFC API.

5.5.4 Resource Links

REST requires resource representations to be linked explicitly to related resources and top-level links among collections in RESTful IFC API is shown in Figure 46. Within the REST metadata, "href" property shown in Table 26 is the URL for where the resource resides and uniquely defines the resource. Also, "link" property identifies a relationship to

another resource that itself has an “href” property to identify the URL for the link and “rel” property which indicates the relationship or the purpose of the link.

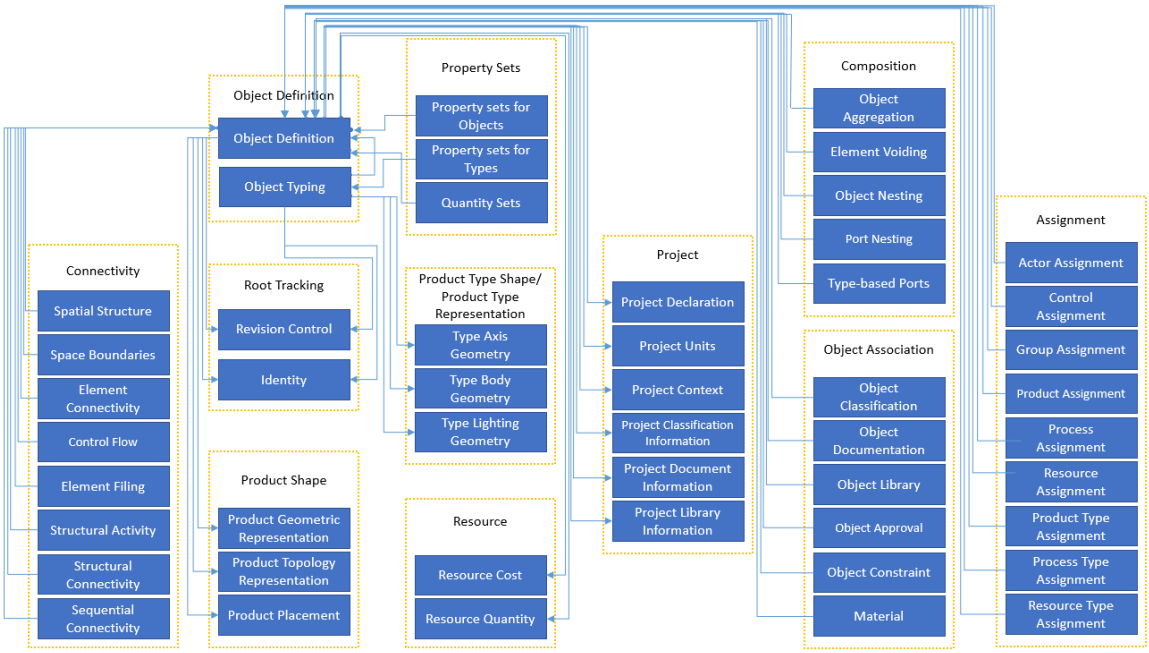


Figure 46 Resource links among top-level collections in IFC REST API

A more conventional design for handling links would be the use of foreign keys for interlinking modularized resources. In this case, resource IDs (i.e. database keys) can be used. Besides, “instanceId” property as discussed in chapter 4 can be used for better readability of IFC files, especially for the purpose of translating BIM data to and from an IFC file directly. Since there are no current sources of data i.e., ifcJSON BIM models available, in this research the IFC files will be translated and transferred to the REST API. So, the “instanceIds” will also be used. This will assist with translating the BIM model back to IFC file for evaluation purposes which will be discussed in chapter 7. GUID or global identification defined in IFC specification falls short in providing REST links because some of the concepts may not include entities with GUID, as will be discussed

later in the next chapter. Semantics for the resource links in this RESTful IFC API follows the logics defined for IFC concepts. If an IFC concept has relationships to another IFC concept, its corresponding REST collection will also be linked to another collection.

An example of links among the collections and resources is shown in Figure 47. Resource schema is also shown for each collection. Links are illustrated with red lines which follow the semantics of IFC specification. In This example, an "ifcelement" collection is linked to three other collection of resources: 1) "revisioncontrol" collection that defines the "ownerHistory" property of the "ifcelement" resources, 2) "productplacement" collection that defines the "objectPlacement" property of the "ifcelement" resources, 3) "productgeometricrepresentation" collection that defines the "representation" property of the "ifcelement" resources.

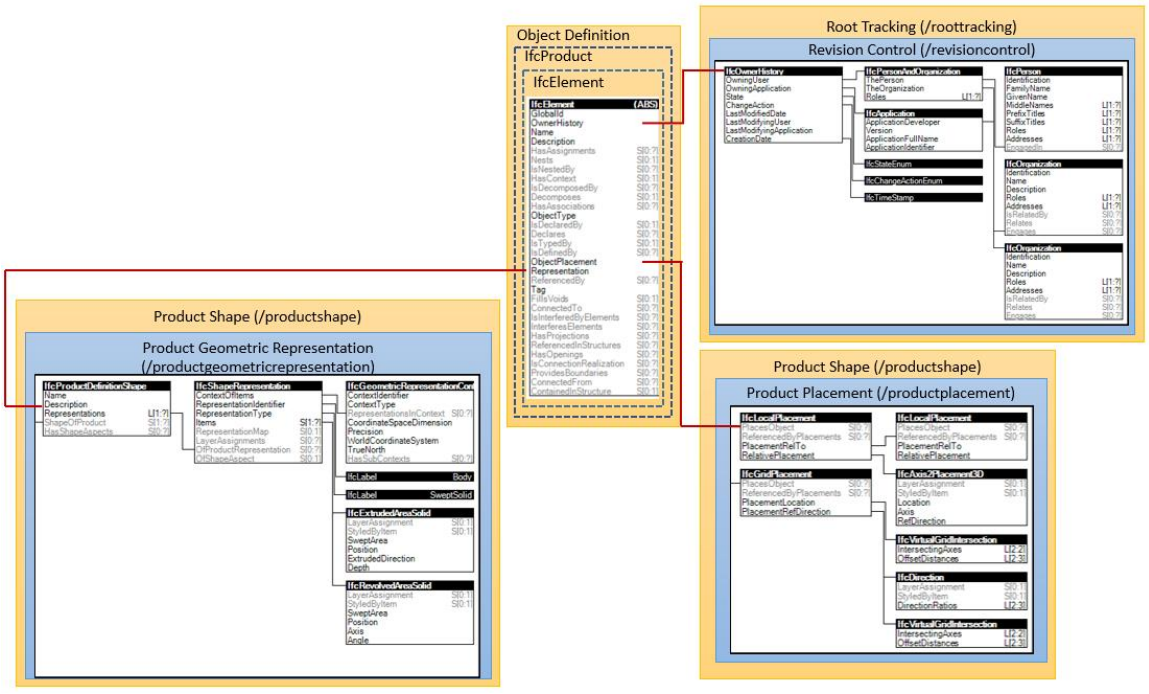


Figure 47 Resource links among four collections

Table 26 shows an example of an API response that returns data for a building element i.e. column. Here, three properties such as “ownerHistory”, “objectPlacement”, and “representation” have links to other resources based on the collection links shown in Figure 47. The “href” property in Table 26 provides the URL to the linked resource that contains the associated data.

Table 26 An ifcJSON resource containing data to represent a column

Raw ifcJSON Data	ifcJSON Data Visualization
{ "_id": "59148b85fb91b217e354d0ff", "instanceId": 259, "ifc": "IFCCOLUMN", "globalId": "3$gxit421D9RMef03b0mxA", "ownerHistory": { "instanceId": 41, "link":{ "rel": "ownerHistory", "href": "http://localhost:3000/ifcrestapi/roottrackings/revisioncontrols/5914903bfb91b217e354d103" } }, "name": "Concrete Column", "description": null, "objectType": "Column", "objectPlacement": { "instanceId": 257, "link":{ "rel": "objectPlacement", "href": "http://localhost:3000/ifcrestapi/productshapes/productplacements/59149138fb91b217e354d104" } }, "representation": { "instanceId": 252, "link":{ "rel": "representation", "href": "http://localhost:3000/ifcrestapi/productshapes/productgeoemtricrepresentations/bodygeometries/bodysweptsolidgeometries/59148f34fb91b217e354d102" } }, "tag": "267108", "predefinedType": "COLUMN"}	{ "_id": "59148b85fb91b217e354d0ff", "instanceId": 259, "ifc": "IFCCOLUMN", "globalId": "3$gxit421D9RMef03b0mxA", "ownerHistory": { "instanceId": 41, "link":{ "rel": "ownerHistory", "href": "http://localhost:3000/ifcrestapi/roottrackings/revisioncontrols/5914903bfb91b217e354d103" } }, "name": "Concrete Column", "description": null, "objectType": "Column", "objectPlacement": { "instanceId": 257, "link":{ "rel": "objectPlacement", "href": "http://localhost:3000/ifcrestapi/productshapes/productplacements/59149138fb91b217e354d104" } }, "representation": { "instanceId": 252, "link":{ "rel": "representation", "href": "http://localhost:3000/ifcrestapi/productshapes/productgeoemtricrepresentations/bodygeometries/bodysweptsolidgeometries/59148f34fb91b217e354d102" } }, "tag": "267108", "predefinedType": "COLUMN" }

5.5.5 HTTP Interaction Semantics

HTTP has been used widely for web-based information delivery and web technologies can deliver data between the HTTP clients and servers [88]. Standard HTTP

operations, GET, POST, PUT and DELETE, [86] are used for invoking each resource. Here, since the focus is on BIM data transfer, the study only considers GET operation. The HTTP GET method is used to read or retrieve a representation of a resource. In RESTful IFC API, GET returns a representation in JSON or specifically ifcJSON format. Upon success, the HTTP response code is 200 (i.e. OK) but in the case of an error, HTTP response returns either a 404 (i.e. NOT FOUND) or 400 (i.e. BAD REQUEST). GET request is used to read data and does not change the data thus, it can be called without the risk of any modification to the data. An example of a GET request is shown below.

```
GET http://www.bimapp.com/ifcrestapi/productshapes/productplacements/
59149138fb91b217e354d104
```

This request can return a response that is shown in Table 27 which is an ifcJSON object that includes data for an object placement which is an absolute placement. The data is structured based on “Product Placement” concept in IFC4.

Table 27 An ifcJSON resource containing placement data

Raw ifcJSON Data	ifcJSON Data Visualization

{"_id":"59149138fb91b217e354d104","instanceId":257,"ifc":"IFCLOCALPLACEMENT","placementRelTo":null,"relativePlacement":{"refDirection":null,"axis":null,"location":{"coordinates":[0,0,0],"ifc":"IFCCARTESIANPOINT","instanceId":6},"ifc":"IFCAXIS2PLACEMENT3D","instanceId":31}}	{
	"_id": "59149138fb91b217e354d104",
	"instanceId": 257,
	"ifc": "IFCLOCALPLACEMENT",
	"placementRelTo": null,
	"relativePlacement": {
	"refDirection": null,
	"axis": null,
	"location": {
	"coordinates": [
	0,
	0,
	0
	],
	"ifc": "IFCCARTESIANPOINT",
	"instanceId": 6
	},
	"ifc": "IFCAXIS2PLACEMENT3D",
	"instanceId": 31
	}
	}

REST APIs make the semantics of requests visible at the HTTP protocol layer. Besides, as this RESTful IFC API is designed based on IFC specification, the data itself as well as the interaction patterns within the data carry the semantics of the domain. The URIs and the HTTP calls follow the structure of IFC concepts and request/response is based on IFC concepts. Using standard specification makes the interaction with the BIM data harmonized among collaborating Cloud-BIM applications.

The API can also capture MVD semantics since the resources follow MVD-specific concepts in addition to common IFC concepts. An example of an MVD concept is "Projection Element" concept in precast concrete MVD discussed previously in Figure 40. An example of GET request to retrieve projectionelements resources is as follows.

```
GET http://www.bimapp.com/ifcrestapi/pcis/projectionelements
```

This HTTP request returns all resources available in projectionelements collection. An example of the API response is shown in Table 28. In the API response, two resources are returned as two precast corbels. The data regarding three properties: ownerHistory, objectPlacement, and representation is referenced to their corresponding resources using instanceIds. In addition, for each ifcProjectionElement entity, there are links to its corresponding data that contains ownerHistory, objectPlacement, and representation property. The links are shown both in "href" links and in instanceIds.

Table 28 An ifcJSON resource containing projectionelements data

Raw ifcJSON Data	ifcJSON Data Visualization
[{"_id":"59175d400ec03afdd6f35990","instanceId":100483,"ifc":"IFCPROJECTIONELEMENT","globalId":"1i8UbE1ST2Bh2lwnI8fGTf","ownerHistory":{"instanceId":41,"link":{"rel":"ownerHistory","href":"http://www.bimapp.com/api/roottracking/231/revisioncontrol/5914903bfb91b217e354d103"}},"name":"Projection","description":null,"objectType":"Precast Corbel","objectPlacement":{"instanceId":100482,"link":{"rel":"objectPlacement","href":"http://www.bimapp.com/api/productshape/322/productplacement/591640305a62d13087c07b2a"}},"representation":{"instanceId":100479,"link":{"rel":"representation","href":"http://www.bimapp.com/api/productshape/321/productgeometricrepresentation/514/bodygeometry/134/bodysweptsolidgeometry/59160d56fb91b217e354d108"}},"tag":"267320","predefinedType":"NOTDEFINED"},{"_id":"59175d5c0ec03afdd6f35991","instanceId":906,"ifc":"IFCPROJECTIONELEMENT","globalId":"2JG0Dg77nE_B84sXRFrL2a","ownerHistory":{"instanceId":41,"link":{"rel":"ownerHistory","href":"http://www.bimapp.com/api/roottracking/231/revisioncontrol/5914903bfb91b217e354d103"}},"name":"Precast Corbel 40x70x30","description":null,"objectType":"Corbel 40x70x3","objectPlacement":{"instanceId":905,"link":{"rel":"objectPlacement","href":"http://www.bimapp.com/api/productshape/322/productplacement/59175ecf0ec03afdd6f35992"}},"representation":{"instanceId":902,"link":{"rel":"representation","href":"http://www.bimapp.com/api/productshape/321/productgeometricrepresentation/514/bodygeometry/134/bodysweptsolidgeometry/591764040ec03afdd6f35993"}},"tag":"317508","predefinedType":"NOTDEFINED"}]	[{ "_id": "59175d400ec03afdd6f35990", "instanceId": 100483, "ifc": "IFCPROJECTIONELEMENT", "globalId": "1i8UbE1ST2Bh2lwnI8fGTf", "ownerHistory": { "instanceId": 41, "link": { "rel": "ownerHistory", "href": "http://www.bimapp.com/api/roottracking/231/revisioncontrol/5914903bfb91b217e354d103" } }, "name": "Projection", "description": null, "objectType": "Precast Corbel", "objectPlacement": { "instanceId": 100482, "link": { "rel": "objectPlacement", "href": "http://www.bimapp.com/api/productshape/322/productplacement/591640305a62d13087c07b2a" } }, "representation": { "instanceId": 100479, "link": { "rel": "representation", "href": "http://www.bimapp.com/api/productshape/321/productgeometricrepresentation/514/bodygeometry/134/bodysweptsolidgeometry/59160d56fb91b217e354d108" } }, "tag": "267320", "predefinedType": "NOTDEFINED" }, { "_id": "59175d5c0ec03afdd6f35991", "instanceId": 906, "ifc": "IFCPROJECTIONELEMENT", "globalId": "2JG0Dg77nE_B84sXRFrL2a", "ownerHistory": { "instanceId": 41, "link": { "rel": "ownerHistory", "href": "http://www.bimapp.com/api/roottracking/231/revisioncontrol/5914903bfb91b217e354d103" } }, "name": "Precast Corbel 40x70x30", "description": null, "objectType": "Corbel 40x70x3", "objectPlacement": { "instanceId": 905, "link": { "rel": "objectPlacement", "href": "http://www.bimapp.com/api/productshape/322/productplacement/59175ecf0ec03afdd6f35992" } }, "representation": { "instanceId": 902, "link": { "rel": "representation", "href": "http://www.bimapp.com/api/productshape/321/productgeometricrepresentation/514/bodygeometry/134/bodysweptsolidgeometry/591764040ec03afdd6f35993" } }, "tag": "317508", "predefinedType": "NOTDEFINED" }]

5.6 Standardized API for BIM Data Transmission

Tight coupling presents a severe impediment to Cloud-BIM collaboration. REST is a stateless architecture, uses the web's existing protocols and technologies. REST scales well because interactions between client and server are primarily loosely coupled. In addition, in REST architecture, BIM data that is produced and consumed becomes separated from the technologies for production and consumption of data thus, making it highly scalable and easy to modify and extend. Here in RESTful IFC API, server can change the resources freely while the client only needs to know the base URI and then, chooses from the IFC-based resources to perform HTTP operations and retrieve BIM data. In REST architecture, the client-server interactions are centered around resources that are assigned individual URIs which adds flexibility. In fact, RESTful IFC API utilizes IFC schema as open BIM standard to inform resource identification, to guide resource representation, and to structure the URI patterns and then, it exchanges the BIM data over a standardized interface as HTTP. This creates a standardized API that can support BIM data exchange in Cloud which is based on web technologies.

RESTful IFC API is based on a set of REST API design patterns and uses ifcJSON for resource representation. This makes the resources compatible with REST as it follows JSON data model and additionally aligns the resources with the AEC standards since it follows IFC specification. The identification of API resources and the resource schema itself is based on IFC concepts and MVDs. As a result, the API resources are standardized for the AEC industry while it also captures domain semantics since MVD efforts are aimed at providing domain-specific semantic clarity for BIM exchanges.

Most importantly, as mentioned IoT requires network connectivity and provision of resources in the application layer on top of the network connectivity through WoT. The RESTful IFC API proposed in this research makes the resources accessible through standard web-based interactions which enable the implementation of IoT and its integration with the BIM process.

The design of IFC REST API resources is not database-driven. Instead, it is based on API interactions and user needs. Next chapter will discuss the implementation of IFC REST API to show how resources can be stored within the persistence layer explained in chapter 6 and illustrated in Figure 52.

CHAPTER 6. EVALUATION OF BIM SYNAPSE

This chapter explains the evaluation process for BIM Synapse framework. The study applies empirical evaluation to appraise the study hypotheses. The components of the empirical evaluation are discussed and evaluation metrics are specified. Since the key to empirical evaluation is the proper design and execution of the experiment, this chapter explains the experiment within application, implementation and test cases. To guide the design of the experiment, the use case application for precast concrete domain is explained. Also, to describe implementation detail of the interoperability framework for Cloud-BIM this chapter specifies the technologies that are used to implement the framework and apply it in the precast concrete use case. In addition, the chapter explains how the test cases for the experiment are generated and applied to guide data validation and evaluation of the framework. The evaluation of BIM Synapse framework is explained in five parts: part 1) empirical evaluation, part 2) application, part 3) implementation, part 4) test cases, part 5) validation and evaluation.

6.1 Part1: Empirical Evaluation

This study uses empirical evaluation methods to evaluate BIM Synapse framework. Empirical studies focus on comparing theory to reality and can be done through experiments, case studies, survey and prototyping [99]. In empirical evaluation, the study hypothesis will be tested and the data will be interpreted to guide the result of the experiment. Empirical studies involve formulating the hypothesis, applying it to use cases, using case studies to abstract requirements into data, analyzing and validating the data, and eventually drawing conclusions based on the hypothesis and research questions [99]. It is

also required to define metrics to strengthen the evaluation. Empirical study components are research context, hypothesis, experimental design, data analysis and validation, results and conclusion [99]. Experimental design and analysis can be done within several steps: conception, design, preparation, execution, analysis, and dissemination [100]. Experimental design and data validation are major components of empirical studies [99] which form the evaluation components. The evaluation process in this study is shown in Figure 48.

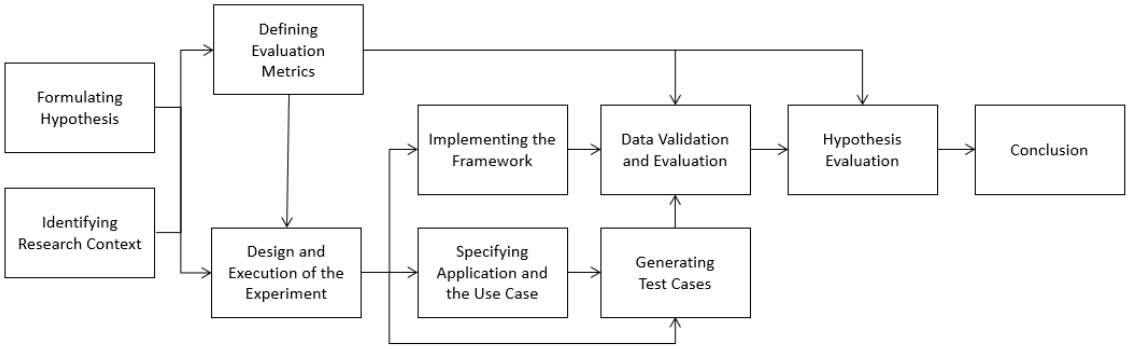


Figure 48 Main stepts in the evaluation process

Previous chapters have discussed the study hypotheses and the research context. This section will focus on the experimental design components such as application, implementation, and test cases, as well as evaluation metrics, and data validation.

6.1.1 Metrics for Evaluation

To guide the evaluation, metrics are needed to measure the experiment and results of the implementation. In the software verification and validation process, four metrics are major evaluation metrics as correctness, accuracy, completeness, and consistency [101]. Four components of evaluation metrics in this study shown previously in Figure 5 are

correctness, accuracy, completeness, and consistency. The evaluation metrics should be selected in a way that they eventually facilitate the evaluation of the study hypothesis and it should specify how the experiment can evaluate the hypothesis [99].

Table 29 Evaluation metrics and criteria description for BIM Synapse framework

Criteria	Description
Correctness	The data that is being used as the base BIM model to be sent to the receiver over BIM Synapse framework should be mapped correctly to web compatible format and it should be valid data.
Accuracy	The implementation of the framework should be accurate and should ensure that the components of the framework are reliable and can perform in an expected way.
Completeness	The framework suggests that by using the proposed architecture a complete BIM model can be received on the receiver side, so, the study should verify whether the received BIM model is a complete BIM model.
Consistency	The study results achieved through using the proposed framework should address the research questions and should indicate that the hypotheses have been met and applied in BIM Synapse.

The evaluation of the proposed framework can be broken into four levels under four criteria that are summarized in Table 29.

a) Checking the correctness of the BIM model being sent. Since the study proposes to translate the BIM model to web compatible data so that they can be sent over the web protocols, the study needs to make sure that the data being used are valid and can represent the actual BIM model. This can be achieved through validation of ifcJSON documents for formatting and additionally against the ifcJSON schema to ensure that the data is correct syntactically and semantically.

b) Checking the accuracy of the framework implementation. The proposed BIM Synapse framework is designed on a RESTful IFC API that manages BIM data request and response, therefore, the study needs to make sure that the implemented framework can perform accurately. This can be achieved by using techniques for REST API testing.

c) Checking the completeness of the received BIM data. The study needs to ensure that the BIM model received using web-based data transfer protocol based on the proposed framework is a complete BIM model. Since there is no available tool or technique to validate the received ifcJSON model, this can be achieved through translating the received ifcJSON data back to IFC STEP file and validating the IFC file using available tools. Also, the framework is using MVD concepts to retrieve modules of the BIM data through REST resources, so the study needs to ensure that the modules of received data correspond to MVD modules. This can be achieved by testing received resources against data that satisfies modularization in related MVD concepts.

d) Checking the consistency of the research outcome with study hypotheses. Since the study proposes the BIM Synapse framework to address the hypotheses, it needs to ensure that the outcome of the procedure can identify the underlying research requirements. This can be achieved by showing how framework components can address the hypothesis and by mapping the evaluation metrics and the study findings with initial research requirements to discuss how the empirical evaluation can result in hypothesis testing.

6.2 Part 2: Application

This study applies the framework for Cloud-BIM interoperability in a precast concrete use case. An example of a BIM model for a precast concrete parking garage is shown in Figure 49.

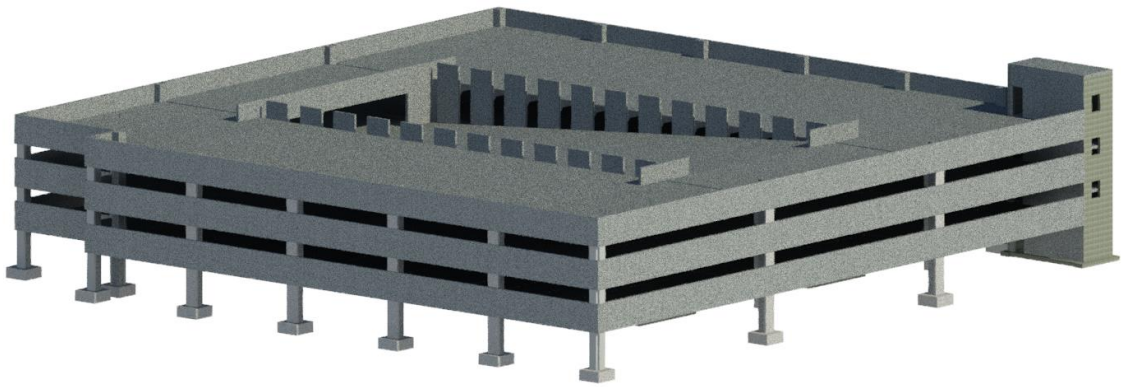


Figure 49 Precast concrete parking garage BIM model

This section first explains the Exchange Model used for the use case within the MVD for precast concrete. Then, it discusses implementation detail of the interoperability framework for Cloud-BIM. It specifies the technologies that are used to implement the framework and applies them in the precast use case.

6.2.1 Precast/Pre-stressed Concrete MVD

Precast Concrete BIM standard [76] defines the specification of the Model View Definition (MVD) for precast model exchanges [77]. Prior to that, Information Delivery Manual (IDM) for Precast Concrete was defined [24]. The IDM specifies all data exchanges for major tasks of the precast fabricator, working with other groups including architects, engineers, general contractors and other sub-contractors such as rebar benders,

proprietary embed fabricators, concrete plants and other procurement specific exchanges [24, 25]. The workgroup formed for the BIM standard effort is divided into four subgroups while each subgroup addressed a different aspect of the precast process for architectural precaster, precaster as lead contractor, precaster as sub-contractor, and fabrication process. Therefore, four sets of Exchange Models were specified by the precast workgroup (shown in Figure 50) including architectural precast exchanges, precaster as lead contractor exchanges, precaster as subcontractor exchanges, and the fabrication backend defining backend production exchanges, with a total of 47 distinct exchanges [24, 25, 102].

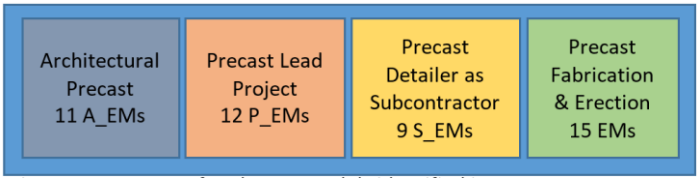


Figure 50 Four sets of Exchange Models identified in precast concrete IDM

The IDM for precast concrete includes 11 exchange models for the architectural precast process (i.e. A_EM), 12 exchange models for precast lead project process (i.e. P_EM), 9 exchange models for precast detailer as subcontractor process (i.e. S_EM), and 15 exchange models for precast fabrication and erection process (i.e. EM) [26, 24]. However, among these Exchange Models there are many similar exchanges [24, 25, 102]. The study in [77] indicated that the existing recommendations for consolidating 47 originally defined exchange models described in the work of [24], [25] and [102] have not been effective. The consolidation methodology proposed in [77] addressed the existing issues with previously defined precast concrete Exchange Models. Thus, 47 originally-defined Exchange Models were combined following the consolidation strategy described

in [77] and twelve exchange models were identified. These 12 Exchange Models are Building Concept, Precast Concept, Precast Contract Development, Engineering Design Development, Architectural Contract, Engineering Contract, Precast Detailed Coordination, Structural Review and Coordination, Engineering Analysis result, Final Precast Detailing and Coordination, Production and Erection Data, and Architectural Review and Coordination. Among these Exchange Models, the EMPC1 which is also known as Building Concept is used in this study.

6.2.2 EMPC.1 and Instance Model

EMPC.1 is a subset of IFC schema that consists of architectural concept model or engineering concept model passed to detailer for further preliminary precast structural and fabrication detailing. The exchange as shown in Table 30, happens at the preliminary stage of the project and it carries concept design layout of precast pieces.

Table 30 EMPC.1 source and recepient

Exchange Models	**Project stage**	**Source**	**Recipient**
EMPC.1 Building Concept	Preliminary Project Description	Architect	Structural Engineer and Precast Detailer
		Structural Engineer	Architect and Precast Detailer

EMPC.1 has Level of Development (LOD) 200 and consists of concept design layout of precast pieces optionally composed into assemblies. Geometry is nominal, without camber or twisting. It does not include surface or structural detailing. It includes structural- and other grid-controls, if used. It optionally includes major architectural

finishes, and site information. It identifies interfaces with other structural elements and curtain wall systems. Extrusion is used as the geometry representation.

The instance model used in this study consists of building elements and includes the resources that respond both to IFC fundamental concepts and MVD-specific concepts. The model is supposed to be exchanged among architect, structural engineer and precast detailer for further detailing in the conceptual phase of the project.

6.2.3 Revised EMPC.1 Concepts

EMPC.1 uses 52 concepts [77] among 93 reusable concepts defined for precast concrete MVD that are specified in IFC 2x3 and implemented in IfcDoc tool [76]. Figure 51 shows concepts defined in precast concrete MVD that are used in EMPC.1. The concepts that are included in EMPC.1 are marked in orange. These concepts are categorized in information groups including Project and Spatial Hierarchy, Primary Building Elements and Types, Reinforcing, Geometry, Components, Material, Extended Building Elements, and Metadata, Status, and Approvals.

Project & Spatial Hierarchy													
PCI-042	PCI-043	PCI-044	PCI-062	PCI-096	MVC-880	MVC-888	MVC-889	MVC-890	MVC-893	MVC-895	PCI-047	PCI-048	PCI-050	PCI-052
Site Contained in Project	Building Contained in Site	Building Storey Contained in Building	spatial structure containment	site curve 2D	Site Attributes	Metric Project Units	Imperial Project Units	Project Name	Building Attributes	Building Storey Attributes	Grid Name	Grid Representation	Grid Axis Assignment	Placement Relative to Grid

Primary Building Elements and Types							Reinforcing					
PCI-054	PCI-063	PCI-064	PCI-067	PCI-081	PCI-154	PCI-155	PCI-072	PCI-103	PCI-104	PCI-107	PCI-118	PCI-133
Element type assign	Rel placement	Abs placement	Precast Piece Mark	Piece Type Geometry Assignment	Precast Topping Attributes	Precast Element Quantities	Precast Rebar Assignment	Aggregation to Cage	Reinforcing Element Association to aggreg	Reinforcing bar attribs	tendon attribs	reinf mesh attribs

Geometry									
PCI-066	PCI-068	PCI-069	PCI-070	PCI-088	PCI-099	PCI-101	MVC-818	MVC-836	MVC-581
Brep geom	Extrude geom	Arbit. Profile	Arbit. Profile w/voids	rebar swept disk	embed type geom	embed geom assign	Face Based Surface Model	Generic Geometric Representation	Root Attribute

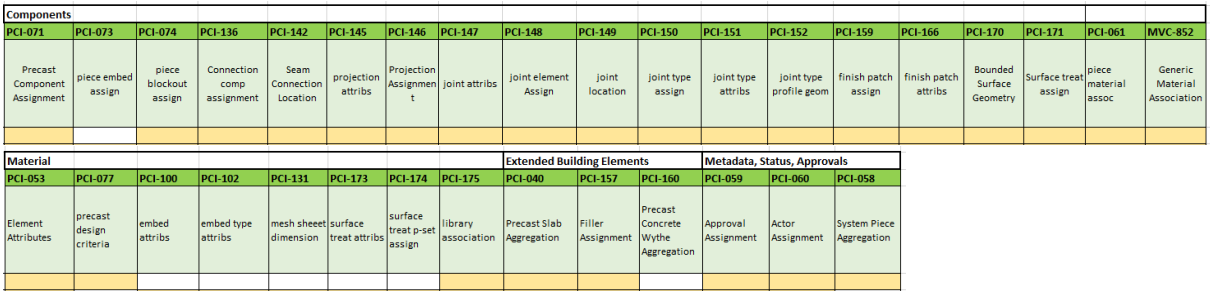
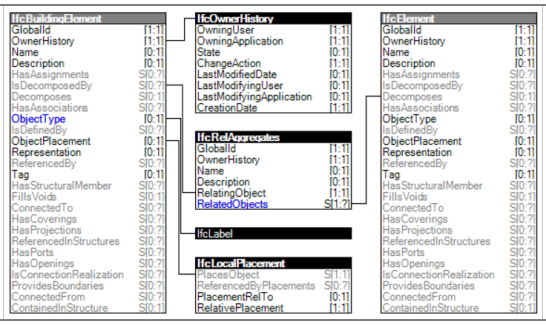
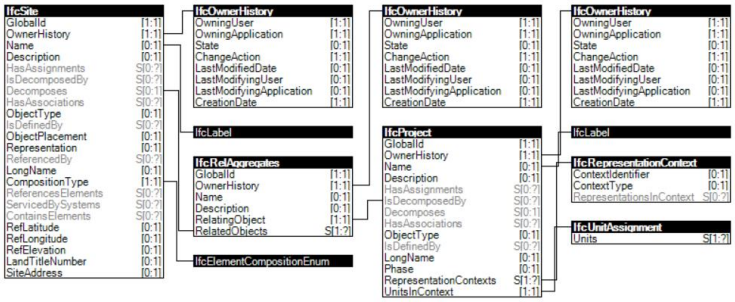
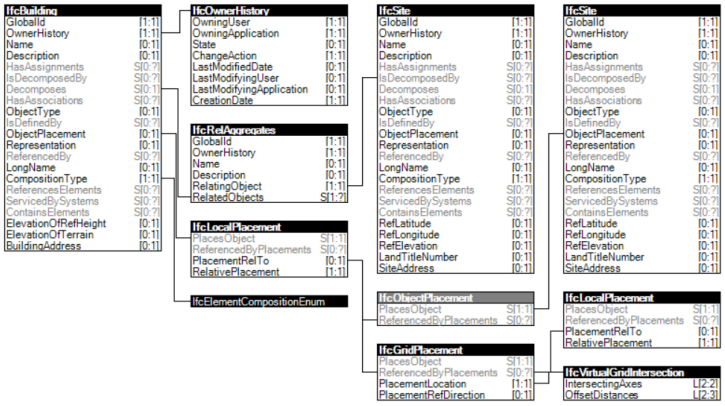
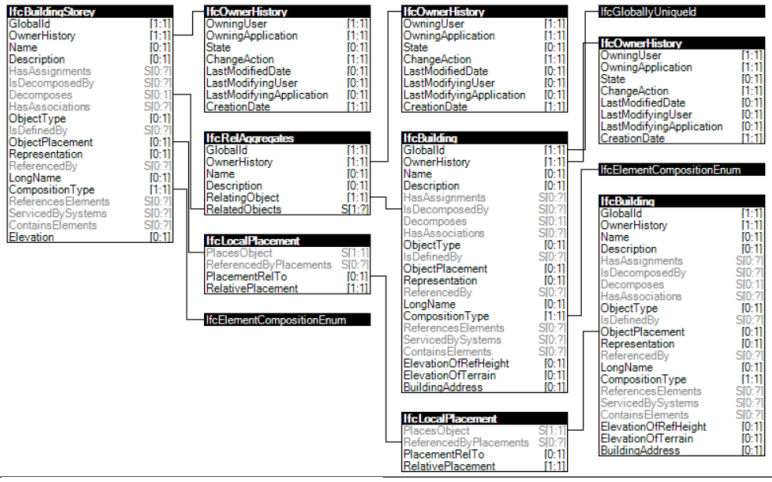
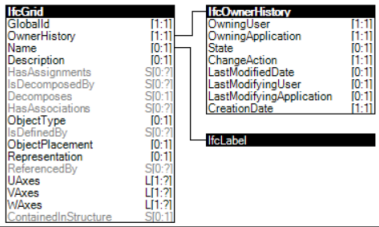
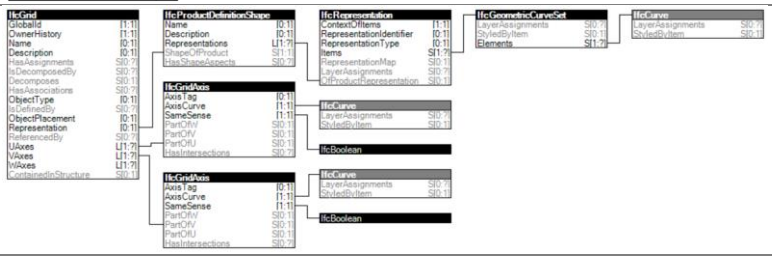


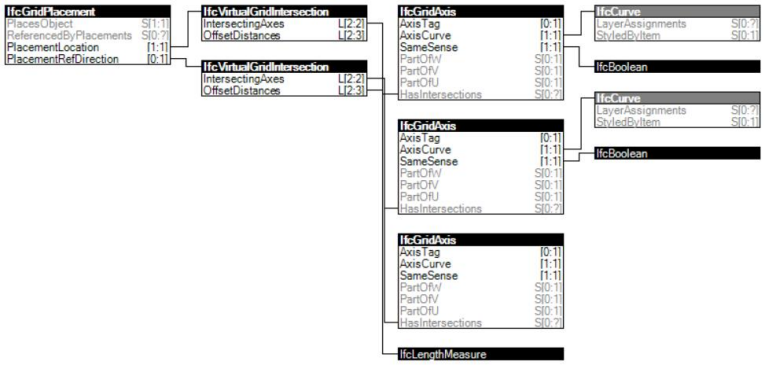
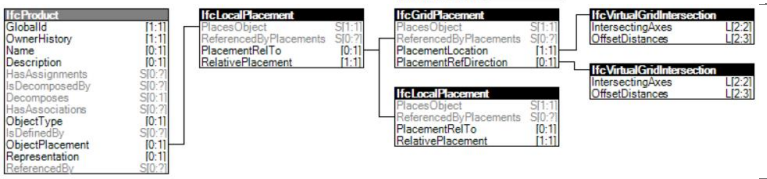
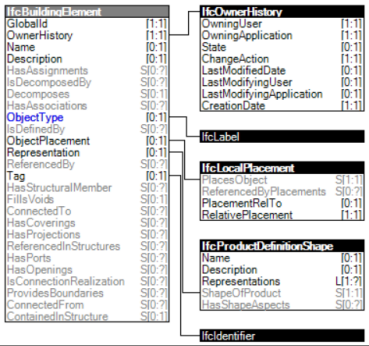
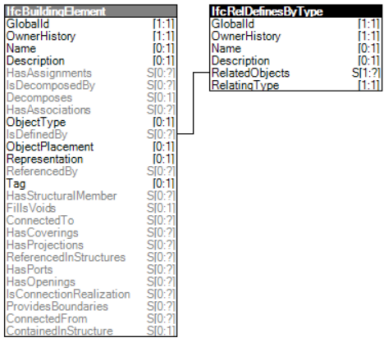
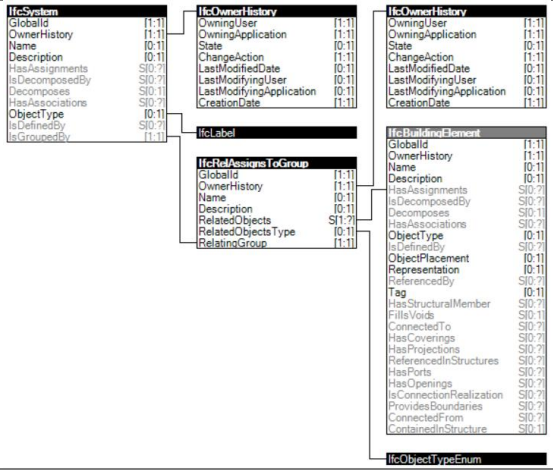
Figure 51 Precast concrete MVD concept taxonomy used in EMPC.1

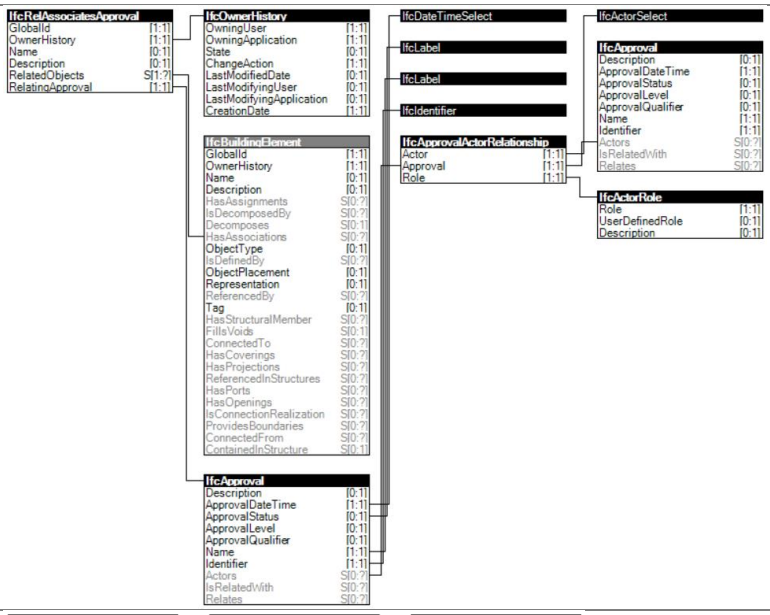
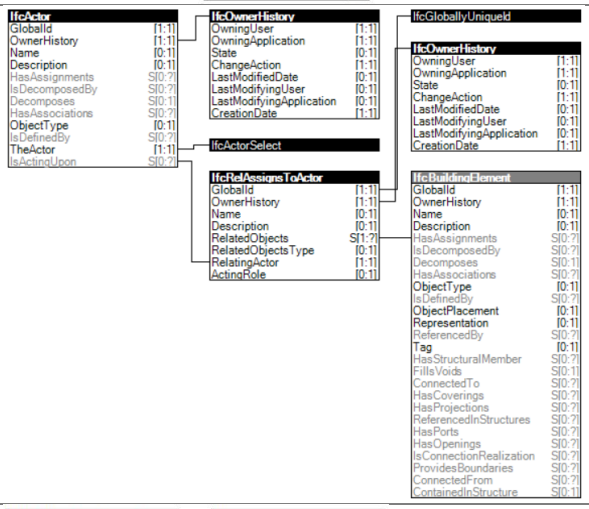
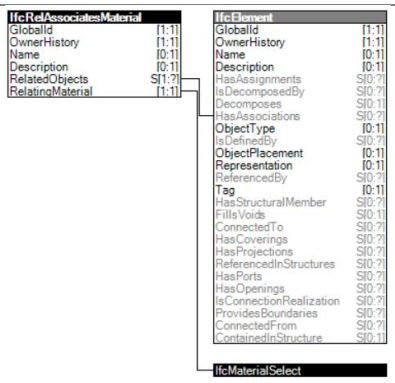
MVD development for Precast Concrete was supported by Precast Concrete Institute (PCI) and the MVD refers to as PCI MVD. The concept definition in PCI MVD follows IFC2x3 specification so, to apply the framework for Cloud-BIM interoperability for EMPC.1 Exchange Model, first PCI MVD needs to be upgraded to IFC4 Add2 which is the current IFC specification. In addition, since IFC4 has specified IFC fundamental concepts, PCI MVD should be revised based on the integration of PCI concepts and IFC4 concepts. Table 31 lists IFC concepts defined in PCI MVD that are used in EMPC.1 Exchange Model.

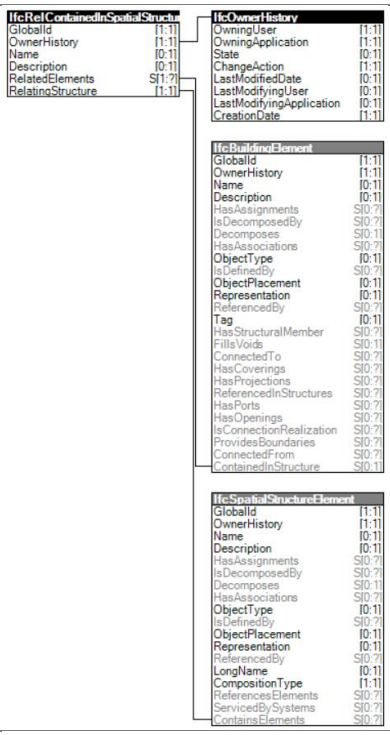
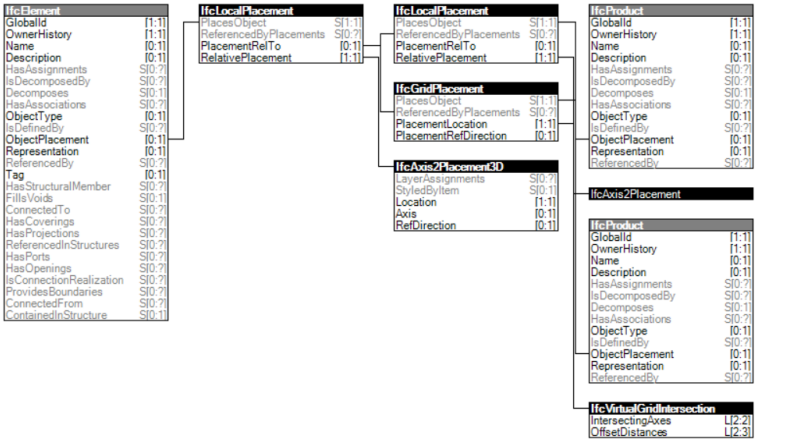
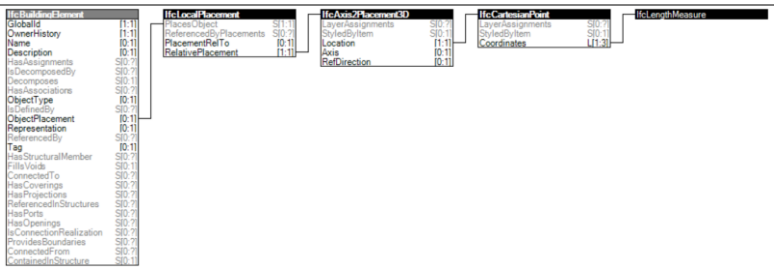
Table 31 List of IFC concepts in MVD for precast concrete

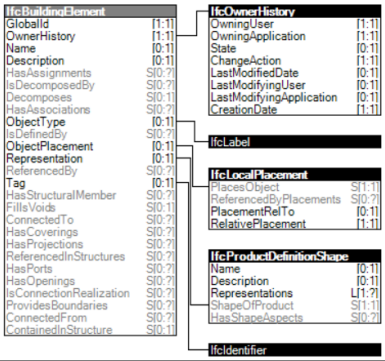
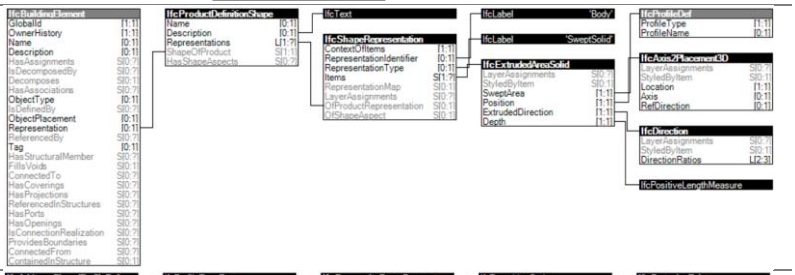
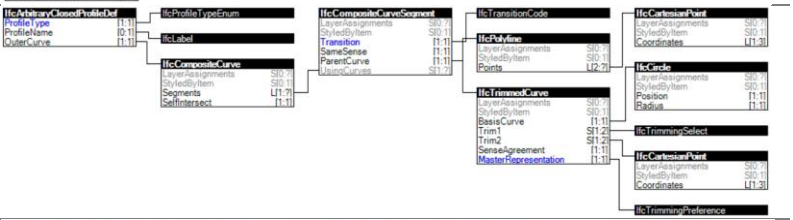
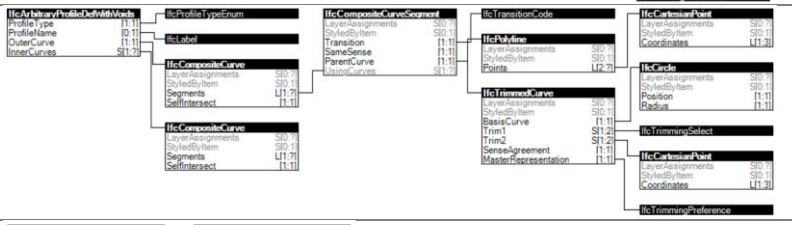
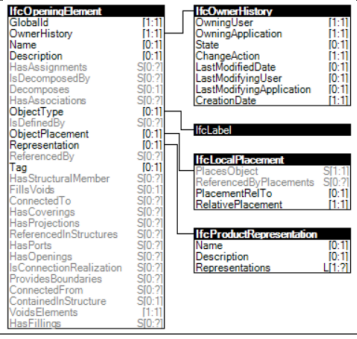
PCI MVD Concepts	Name	Concept Diagram
PCI-040	Building Element Aggregation	

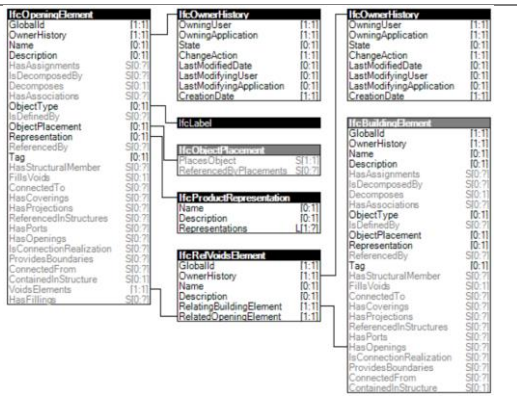
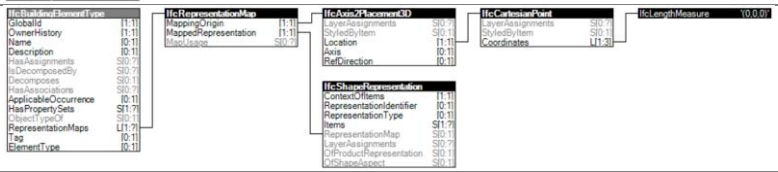
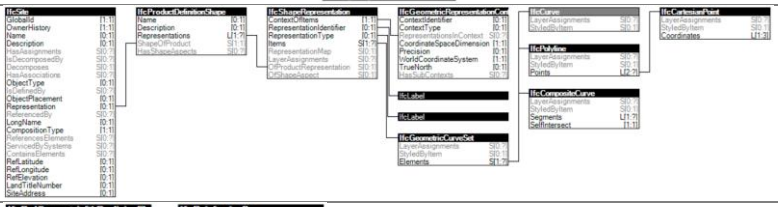
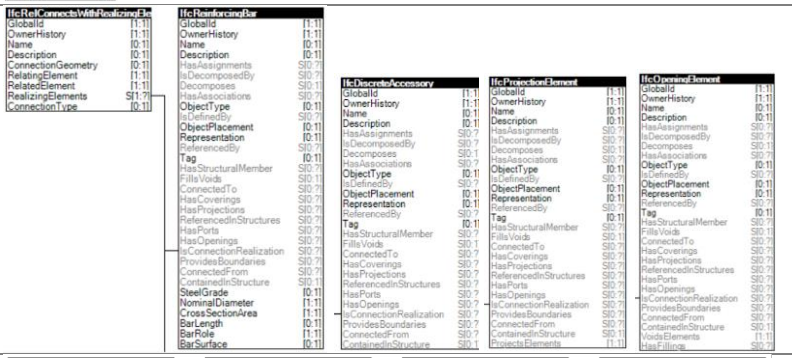
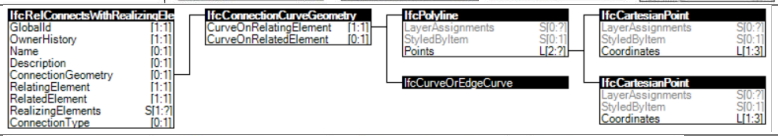
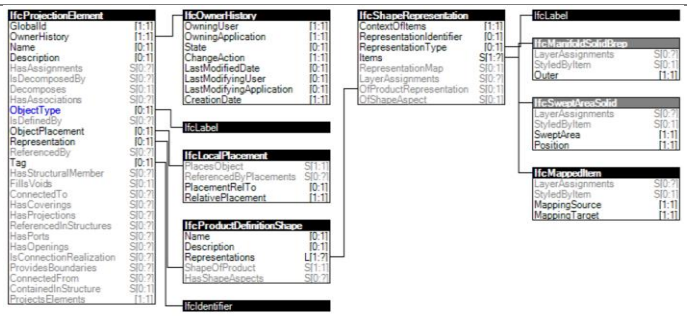
PCI-042	Site Contained in Project	
PCI-043	Building Contained in Site	
PCI-044	Building Storey Contained in Building	
PCI-047	Grid Name	
PCI-048	Grid Representation	

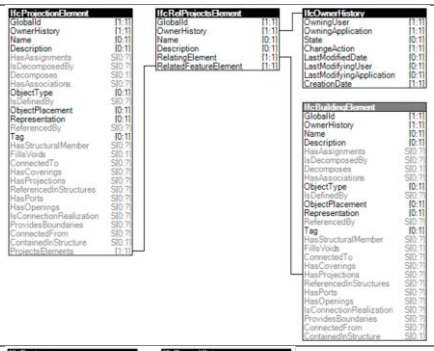
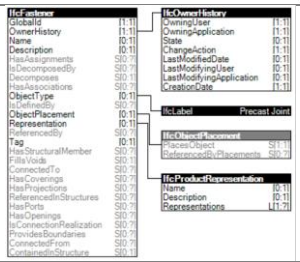
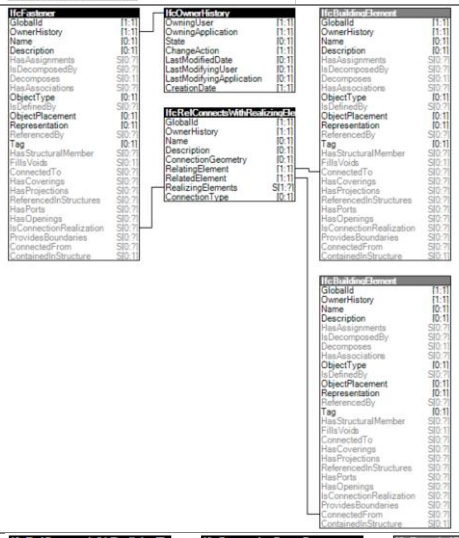
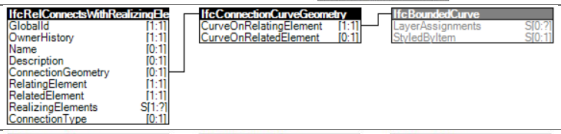
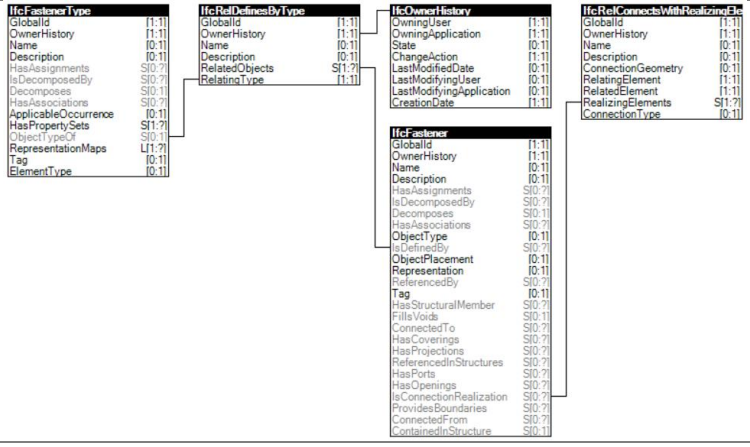
PCI-050	Grid Axis Assignment	
PCI-052	Object Placement Relative to Grid	
PCI-053	Building Element Attributes	
PCI-054	Building Element Type Assignment	
PCI-058	Aggregation of Pieces in the System	

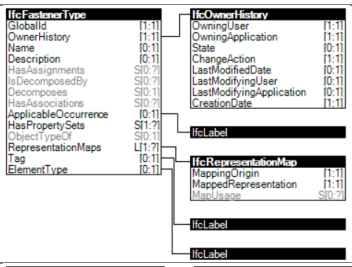
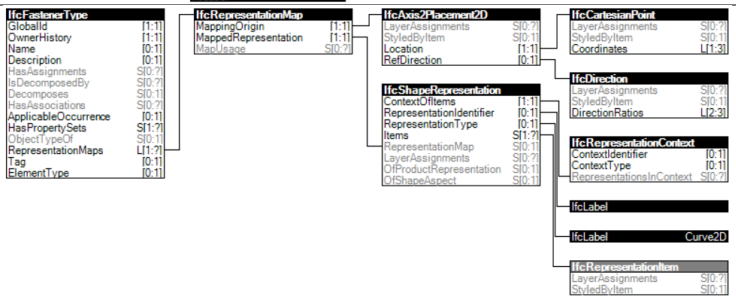
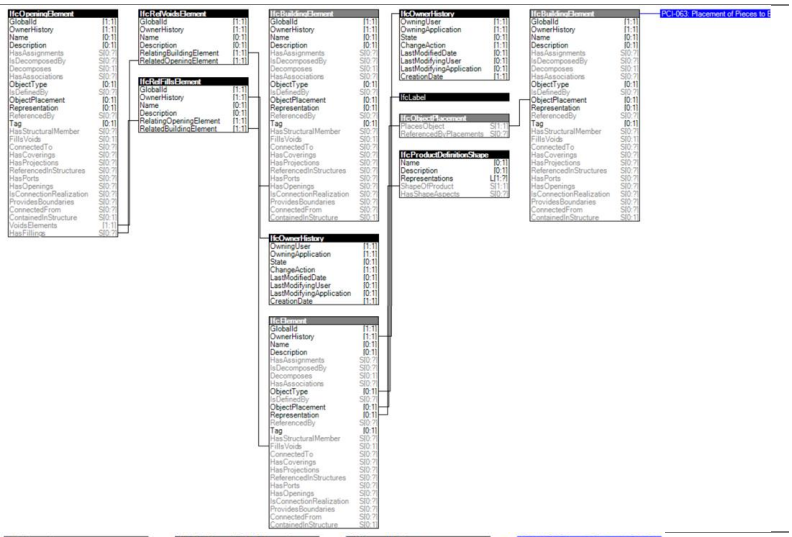
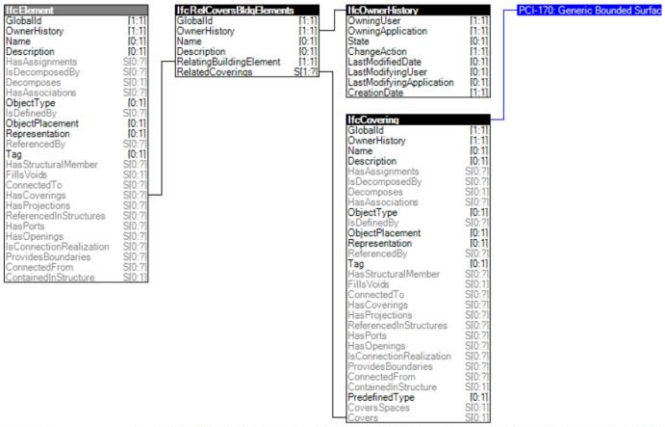
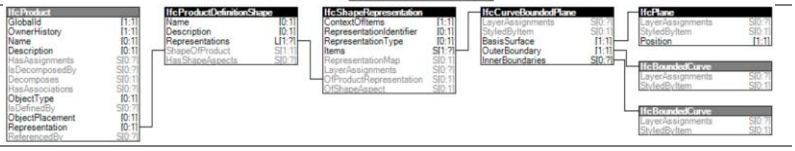
PCI-059	Assignment of Approval	
PCI-060	Assignment of Actor	
PCI-061	Associate Material to Piece	

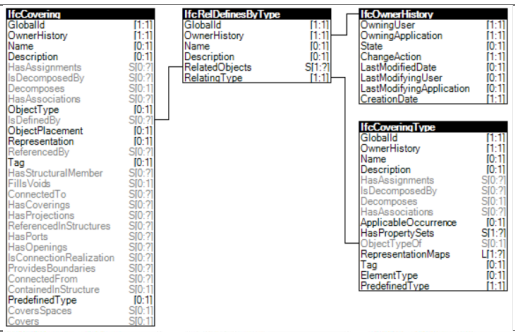
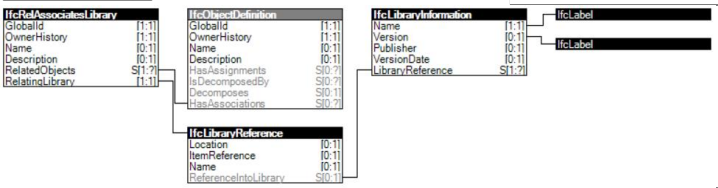
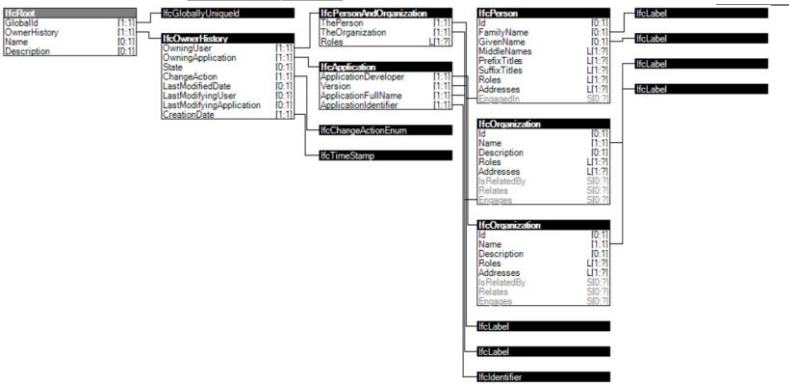
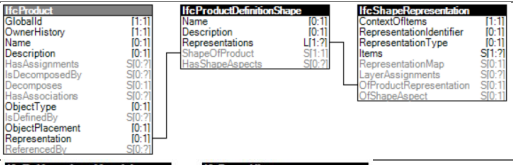
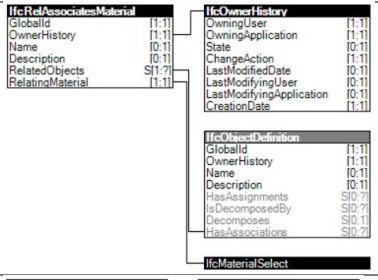
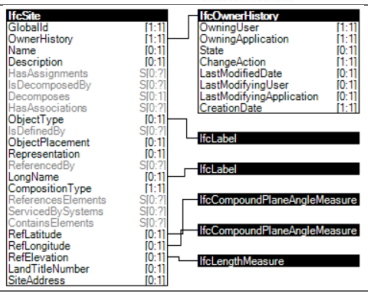
PCI-062	Building Element Assignment to Spatial Structure	
PCI-063	Placement of Pieces to Building Element	
PCI-064	Absolute Placement of Building Elements	

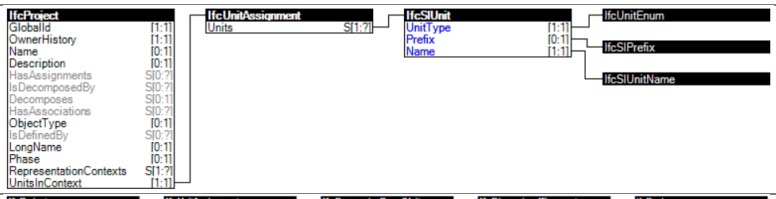
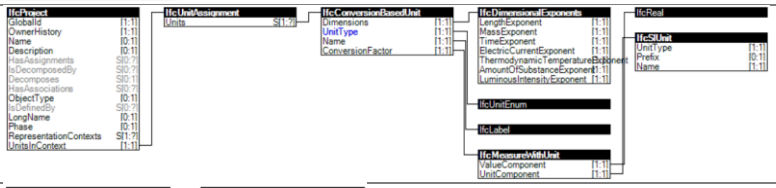
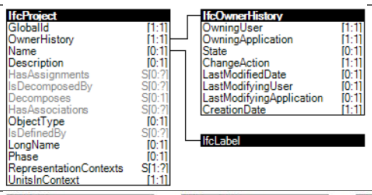
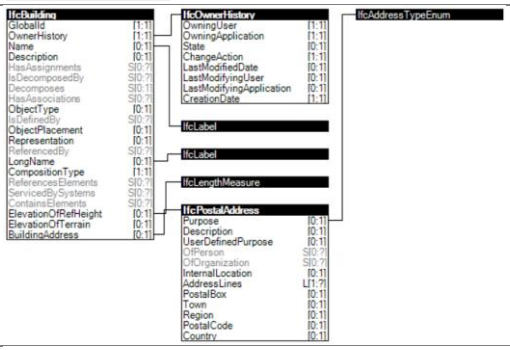
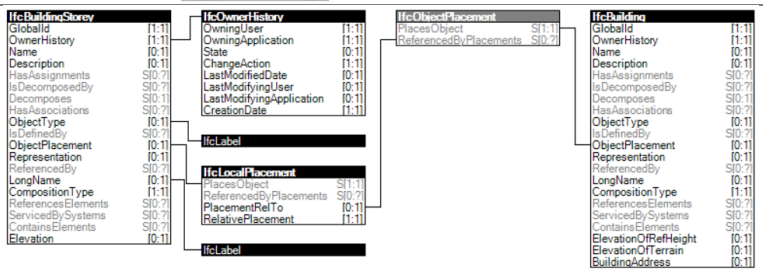
PCI-067	Piece Mark of Building Element	
PCI-068	Extruded Shape Geometry	
PCI-069	Arbitrary Profile	
PCI-070	Arbitrary Profile with Voids	
PCI-071	Precast Blockout Attributes	

PCI-074	Precast Blockout Assignment	
PCI-077	Precast Design Properties	
PCI-081	Building Element Type Representation	
PCI-096	Site Geometric Shape Representation	
PCI-136	Precast Connection Component Assignment	
PCI-142	Precast Seam Connection Location	
PCI-145	Precast Projection Attributes	

PCI-146	Precast Projection Element Assignment	
PCI-147	Precast Joint Attributes	
PCI-148	Precast Joint Element Assignment	
PCI-149	Precast Joint Location	
PCI-150	Precast Joint Type Assignment	

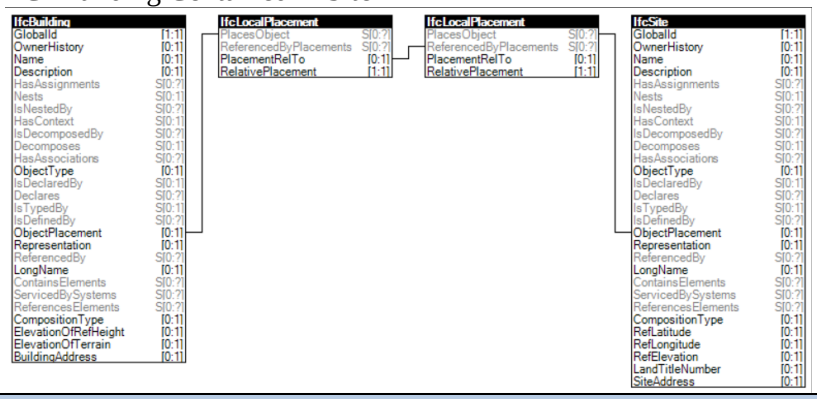
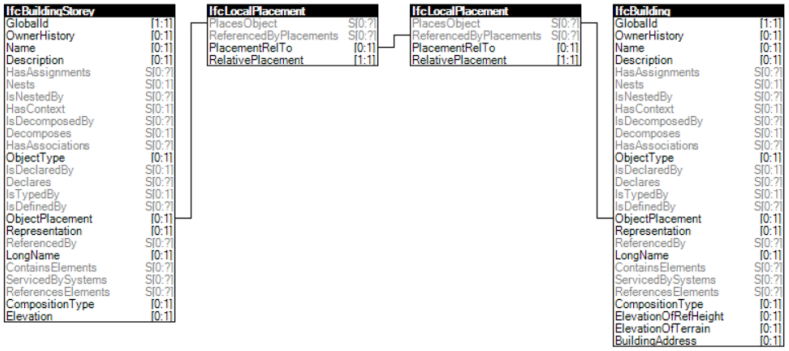
PCI-151	Precast Joint Type Attributes	
PCI-152	Precast Joint Type Representation	
PCI-157	Filler Assignment	
PCI-159	Precast Surface Treatment Assignment	
PCI-170	Generic Bounded Surface Geometry	

PCI-171	Precast Surface Treatment Assignment	
PCI-175	Library Association	
MVC-581	Root Attributes	
MVC-818	Faced Based Surface Model	
MVC-836	Generic Geometric Representation	
MVC-852	Generic Material Association	
MVC-880	Site Attributes	

MVC-888	Metric Project Units	
MVC-889	Imperial Project Units	
MVC-890	Project Name	
MVC-893	Building Attributes	
MVC-895	Building Storey Attributes	

In addition to the need for upgrading PCI MVD to IFC4 specification as well as the need for integration of PCI concepts with IFC4 concepts, analysis of the PCI concepts in Table 31 shows many inconsistencies. For instance, there are many redundant modules that are replicated in many concepts such as IfcOwnerHistory in PCI-040, PCI-042, PCI-151, MVC-895, etc. while this module has been managed in one IFC4 concept in Root Tracking/ Revision Control. Table 32 shows revised PCI concepts and their mapping to IFC4 concepts.

Table 32 List of revised IFC concepts for precast concrete MVD

PCI MVD Concepts	Revised Concepts
PCI-040	IFC4 Concepts
	Composition/Element Composition Root Tracking/Revision Control Product Shape/Product Placement
	Revised PCI Concept
	-
PCI-042	IFC4 Concepts
	Composition/Object Aggregation/Spatial Composition Project/Project Context Root Tracking/Revision Control Project/Project Units
	Revised PCI Concept
	-
PCI-043	IFC4 Concepts
	Composition/Object Aggregation/Spatial Composition Root Tracking/Revision Control
	Revised PCI Concept
	PCI Building Contained in Site 
PCI-044	IFC4 Concepts
	Composition/Object Aggregation/Spatial Composition Root Tracking/Revision Control
	Revised PCI Concept
	PCI Building Storey Contained in Building 

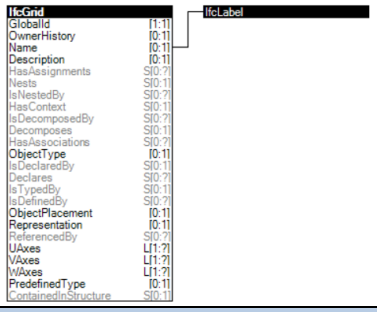
PCI-047

IFC4 Concepts

Root Tracking/Revision Control

Revised PCI Concept

PCI Grid Name



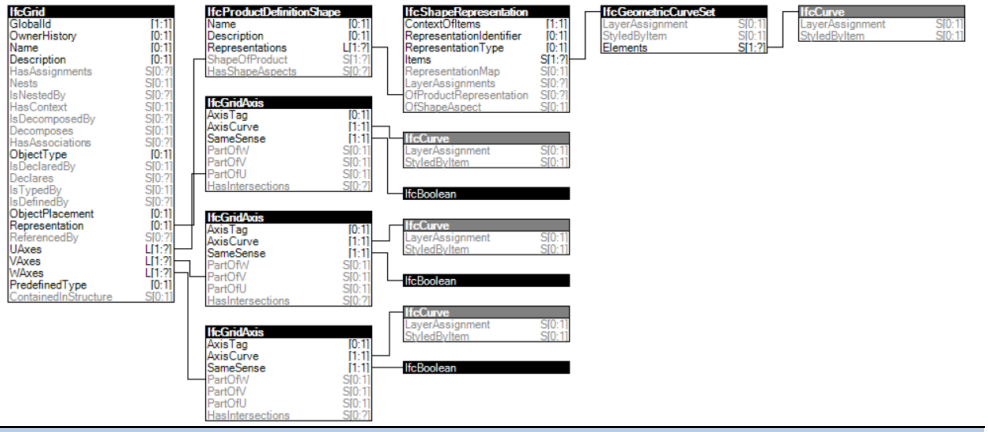
PCI-048

IFC4 Concepts

-

Revised PCI Concept

PCI Grid Representation



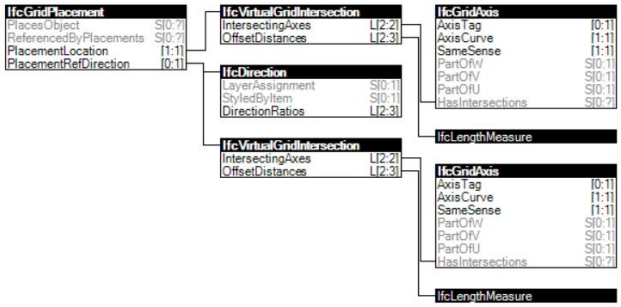
PCI-050

IFC4 Concepts

-

Revised PCI Concept

PCI Grid Axis Assignment



PCI-052

IFC4 Concepts

Product Shape/Product Placement

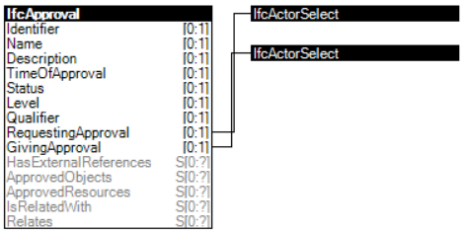
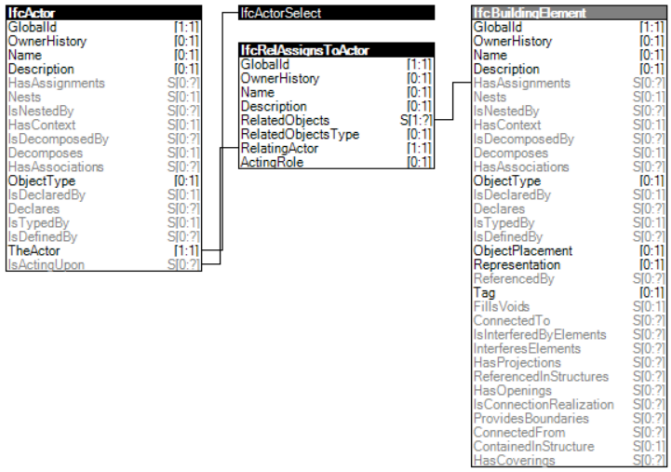
Revised PCI Concept

-

PCI-053

IFC4 Concepts

Root Tracking/Revision Control
Product Shape/Product Placement

	Product Shape/Product Geometric Representation
	Revised PCI Concept
	-
PCI-054	IFC4 Concepts
	Object Definition/Object Typing
	Revised PCI Concept
	-
PCI-058	IFC4 Concepts
	Assignment/Group Assignment Root Tracking/Revision Control
	Revised PCI Concept
	-
PCI-059	IFC4 Concepts
	Object Association/Object Approval
	Revised PCI Concept
	PCI Assignment of Approval 
PCI-060	IFC4 Concepts
	Root Tracking/Revision Control Root Tracking/Identity
	Revised PCI Concept
	PCI Assignment of Actor 
PCI-061	IFC4 Concepts
	Object Association/Material/Material Single
	Revised PCI Concept
	-
PCI-062	IFC4 Concepts
	Root Tracking/Revision Control Connectivity/Spatial Structure/Spatial Containment
	Revised PCI Concept

-

PCI-063

IFC4 Concepts

Product Shape/Product Placement

Revised PCI Concept

-

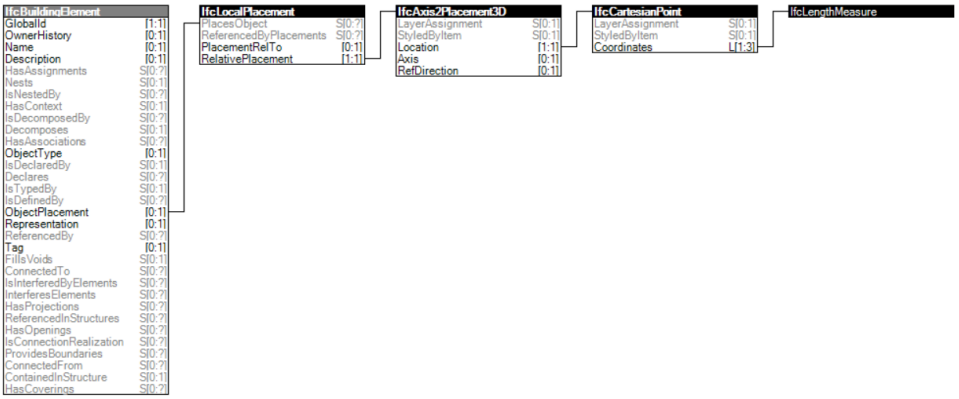
PCI-064

IFC4 Concepts

-

Revised PCI Concept

PCI Absolute Placement of Building Elements



PCI-067

IFC4 Concepts

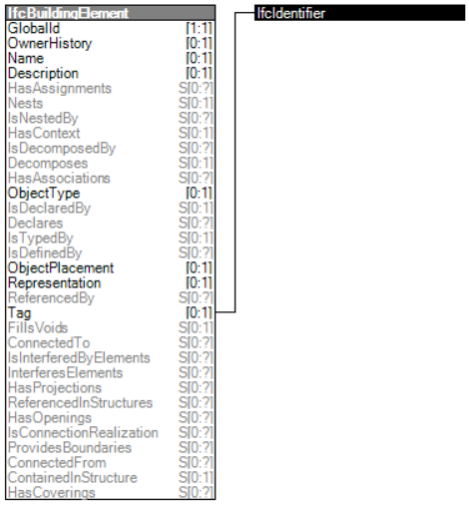
Root Tracking/Revision Control

Product Shape/Product Placement

Product Shape/Product Geometric Representation

Revised PCI Concept

PCI Piece Mark of Building Element



PCI-068

IFC4 Concepts

Product Shape/Product Geometric Representation/Body Geometry/Body SweptSolid Geometry

Revised PCI Concept

-

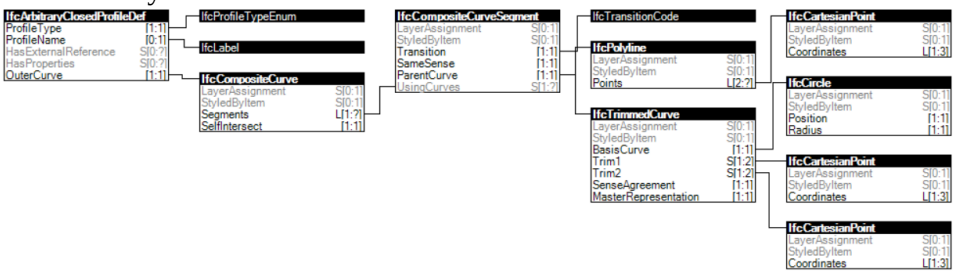
PCI-069

IFC4 Concepts

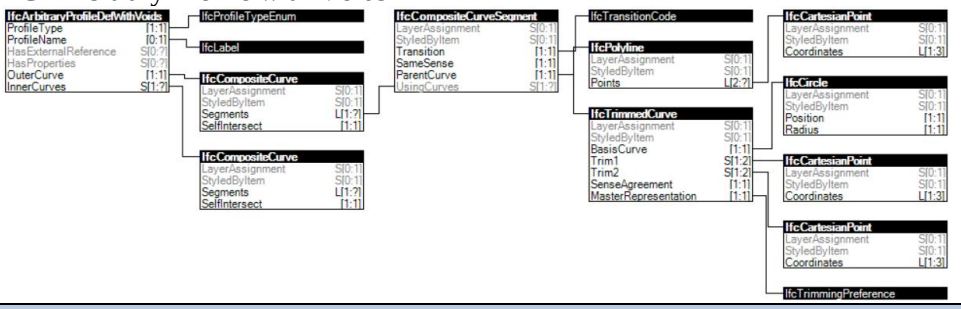
-

Revised PCI Concept

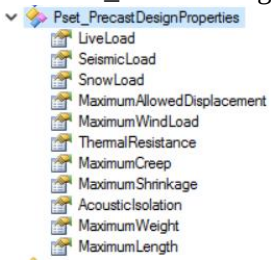
PCI Arbitrary Profile



PCI-070	IFC4 Concepts
	-
	Revised PCI Concept
	PCI Arbitrary Profile with Voids



PCI-071	IFC4 Concepts
	Root Tracking/Revision Control Product Shape/Product Placement Product Shape/Product Geometric Representation
	Revised PCI Concept
	-
PCI-074	IFC4 Concepts
	Root Tracking/Revision Control Product Shape/Product Placement Product Shape/Product Geometric Representation Composition/Element Voiding
	Revised PCI Concept
	-
PCI-077	IFC4 Concepts
	Object Definition/Property Sets for Objects
	Revised PCI Concept
	PCI Pset_PrecastDesignProperties



PCI-081	IFC4 Concepts
	Product Type Shape/Type Body Geometry
	Revised PCI Concept
	-

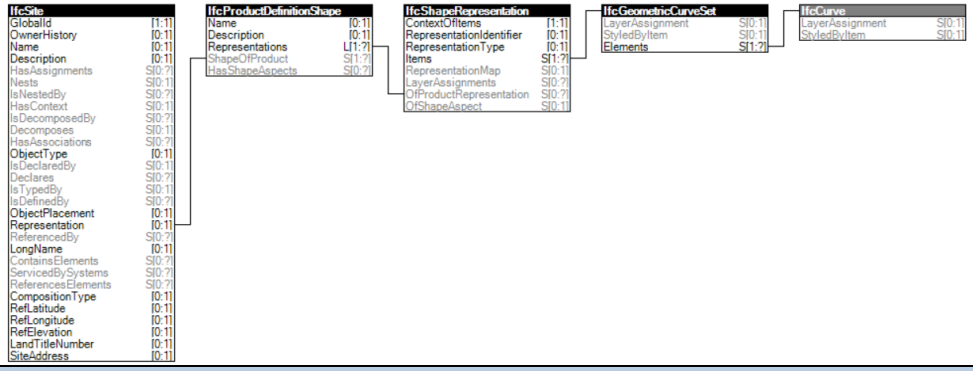
PCI-096

IFC4 Concepts

Project/Project Context

Revised PCI Concept

PCI Site Geometric Curve Representation



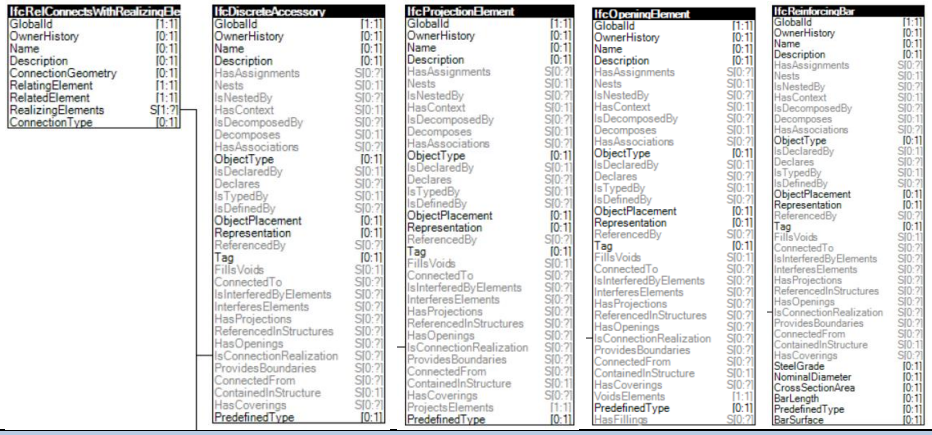
PCI-136

IFC4 Concepts

-

Revised PCI Concept

PCI Precast Connection Component Assignment



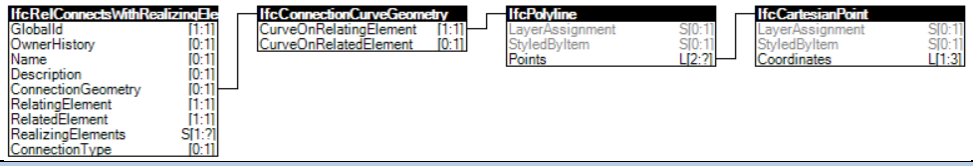
PCI-142

IFC4 Concepts

-

Revised PCI Concept

PCI Precast Seam Connection Reaction



PCI-145

IFC4 Concepts

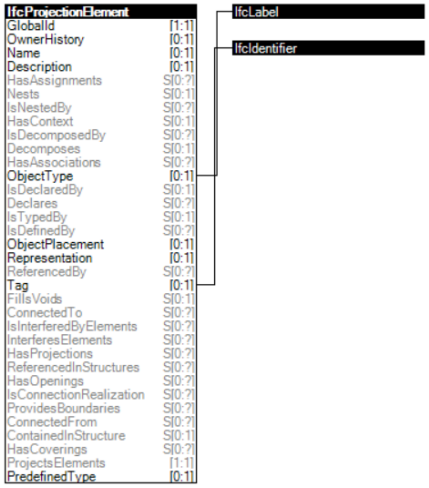
Root Tracking/Revision Control

Product Shape/Product Placement

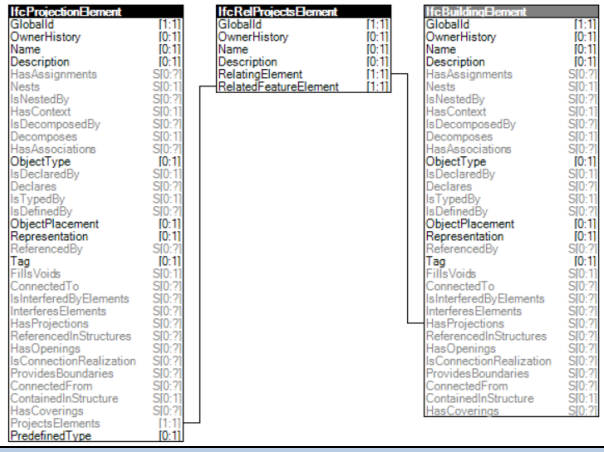
Product Shape/Product Geometric Representation

Revised PCI Concept

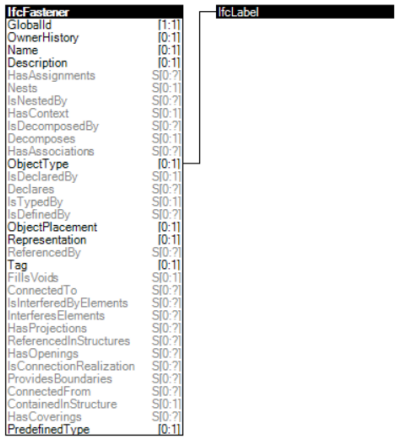
PCI Precast Projection Attributes



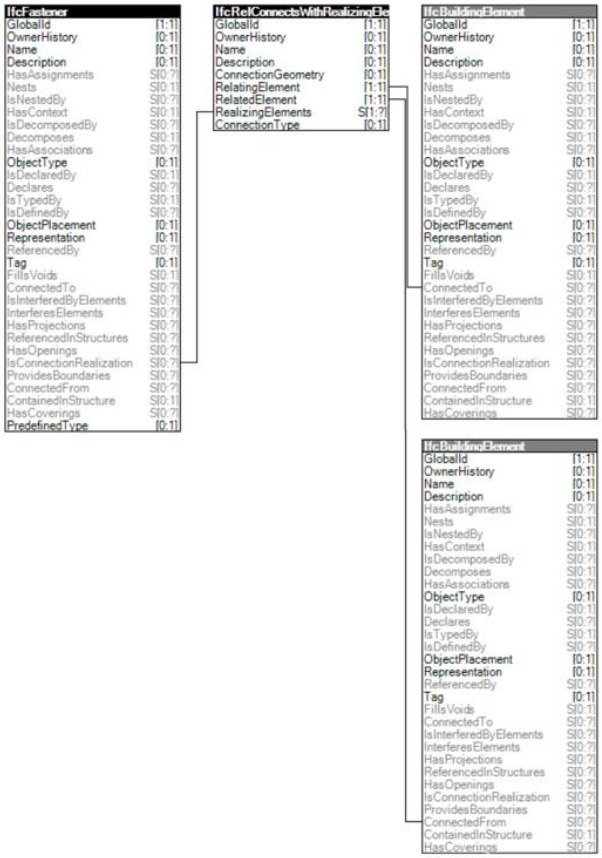
PCI-146	IFC4 Concepts
	Root Tracking/Revision Control
	Revised PCI Concept
	PCI Precast Projection Element Assignment



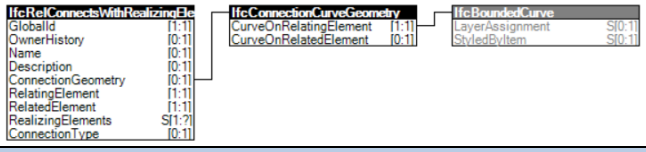
PCI-147	IFC4 Concepts
	Root Tracking/Revision Control
	Product Shape/Product Placement
	Product Shape/Product Geometric Representation
	Revised PCI Concept
	PCI Precast Joint Attributes



PCI-148	IFC4 Concepts
	Root Tracking/Revision Control
	Revised PCI Concept
	PCI Precast Joint Element Assignment



PCI-149	IFC4 Concepts
	-
	Revised PCI Concept
	PCI Precast Joint Location



PCI-150	IFC4 Concepts
	Object Definition/Object Typing
	Revised PCI Concept
	-
PCI-151	IFC4 Concepts
	Object Definition/Object Typing
	Revised PCI Concept
	-
PCI-152	IFC4 Concepts
	Product Type Shape/Type Body Geometry
	Revised PCI Concept
	-

PCI-157

IFC4 Concepts

Root Tracking/Revision Control
Product Shape/Product Placement
Product Shape/Product Geometric Representation
Connectivity/Element Filling
Composition/Element Voiding

Revised PCI Concept

-

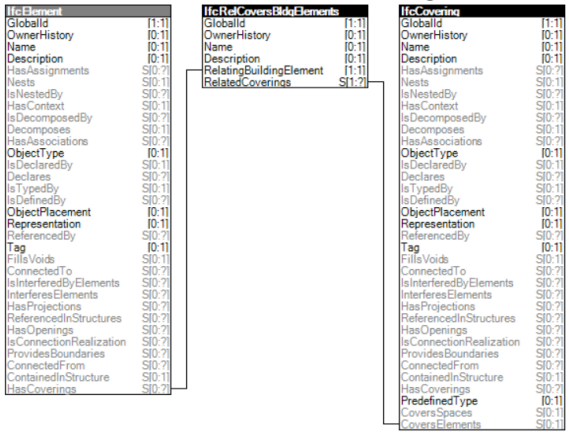
PCI-159

IFC4 Concepts

Root Tracking/Revision Control

Revised PCI Concept

PCI Precast Surface Treatment Assignment



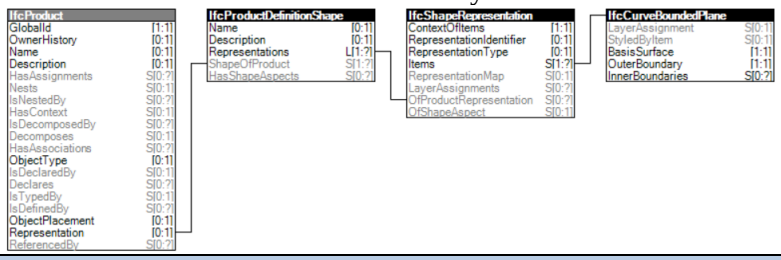
PCI-170

IFC4 Concepts

-

Revised PCI Concept

PCI Generic Bounded Surface Geometry



PCI-171

IFC4 Concepts

Root Tracking/Revision Control
Object Definition/Object Typing

Revised PCI Concept

-

PCI-175

IFC4 Concepts

Object Association/Object Library

Revised PCI Concept

-

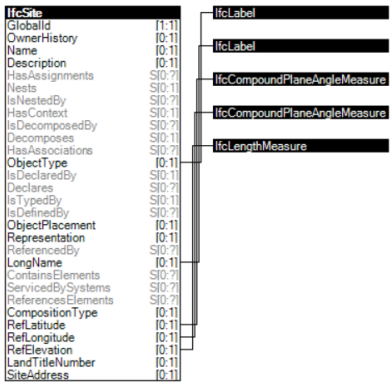
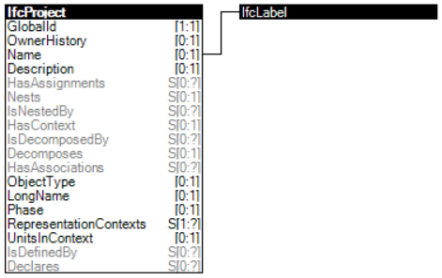
MVC-581

IFC4 Concepts

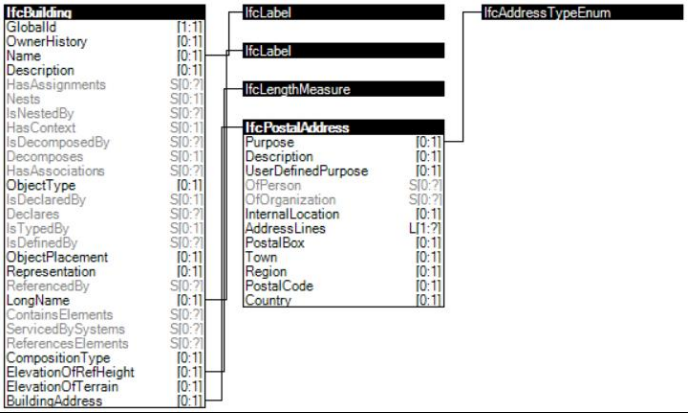
Root Tracking/Identity
Root Tracking/Revision Control

Revised PCI Concept

-

MVC-818	IFC4 Concepts
	Product Shape/Product Geometric Representation/Body Geometry/Body SurfaceModel Geometry
	Revised PCI Concept
	-
MVC-836	IFC4 Concepts
	Product Shape/Product Geometric Representation
	Revised PCI Concept
	-
MVC-852	IFC4 Concepts
	Root Tracking/Revision Control Object Association/Material/Material Single
	Revised PCI Concept
	-
MVC-880	IFC4 Concepts
	Root Tracking/Revision Control
	Revised PCI Concept
	PCI Site Attributes 
MVC-888	IFC4 Concepts
	Project/Project Units
	Revised PCI Concept
	-
MVC-889	IFC4 Concepts
	Project/Project Units
	Revised PCI Concept
	-
MVC-890	IFC4 Concepts
	Root Tracking/Revision Control
	Revised PCI Concept
	PCI Project Name 

MVC-893	IFC4 Concepts
	Root Tracking/Revision Control
	Revised PCI Concept
	PCI Building Attributes



MVC-895	IFC4 Concepts
	Root Tracking/Revision Control Product Shape/Product Placement
	Revised PCI Concept
	PCI Building Storey Attributes

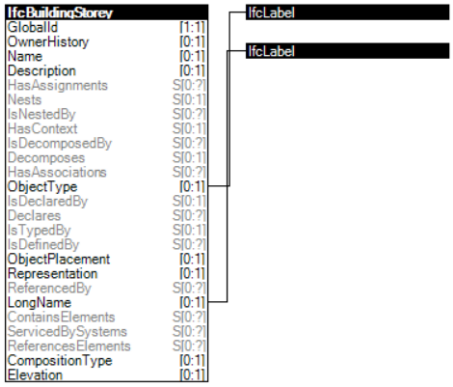


Table 32 captures the inconsistencies and duplicates in PCI concepts by mapping it to IFC4 concepts. For instance, in PCI-150 concept, the IfcRelConnectsWithRealizingElement entity is duplicated while also implemented in PCI-148 therefore, this becomes redundant in PCI-150. Similarly, PCI-061 and MVC-852 concepts are basically the same concept associating material to objects and pieces. Also, the definition of MVC-818 concept has listed obvious mapping of IfcFaceBasedSurfaceModel and based on IFC schema it does not need additional detail.

The upgrade to IFC4 includes revised IFC entities. For example, IfcApproval entity has been changed in IFC4 specification; so, PCI-059 concepts must be revised based on the new definition. All similarities and distinction between PCI concepts and IFC4 concepts are captured in Table 32. Based on this analysis, some of the PCI concepts must be kept but they need to be revised to IFC4 specification. For example, PCI-060 specifies the assignment of actor. While there is a similar concept in IFC4 specification as "Actor Assignment" under "Assignment" concept group but the definition of this concept is different in the PCI concept. In IFC4 specification "Actor Assignment" is dealing with actor assignments through IfcControl such as a work order assigned to an actor or organization [60]. But in the PCI MVD, the definition for PCI-060 concept provides a reference to an actor (either a person or an organization) to building elements including precast and non-precast pieces [76]. So, PCI-060 cannot be mapped to IFC4 "Actor Assignment" concept and needs to be implemented separately. Therefore, in Table 32, when a mapping to IFC4 concepts was possible, they are listed under "IFC4 concepts" and if PCI concept cannot be mapped to IFC4 concepts or there is remaining part of a PCI concept that should be kept separate, it is listed under "Revised PCI Concept" and the concept diagram is illustrated.

Revised concept definition for EMPC.1 has 49 IFC concepts in total including 23 IFC4 fundamental concepts and 26 revised PCI concepts (shown in Table 33).

6.2.4 REST Resources for EMPC.1

As explained in chapter 5, in the framework for Cloud-BIM interoperability, REST resources follow IFC specification. In this study, for implementing the framework for PCI

MVD data exchanges, revised PCI concepts are applied. REST resources correspond to both IFC4 concepts and MVD-specific concepts for precast concrete. As mentioned earlier, these PCI concepts are the concepts used in EMPC.1 Exchange Model. REST resources based on revised PCI concepts are listed in Table 33.

Table 33 REST resources for EMPC.1

IFC Concepts	REST Main Collection	REST Sub-Collections	URI
Project	Project		/projects
Project Units		Project Units	/projectunits
Project Context		Project Context	/projectcontexts
Object Definition	Object Definition		/objectdefinitions
Object Definition		Object Definition	/objectdefinitions
Object Typing		Object Typing	/ objecttypings
Object Definition/ Property Sets	Property Sets		/propertysets
Property Sets for Objects		Property Sets for Objects	/propertysetsforobjects
Object Association	Object Association		/objectassociations
Object Library		Object Library	/objectlibraries
Object Approval		Object Approval	/objectapprovals
Material Single		Material Single	/materialsingles
Product Shape	Product Shape		/productshapes
Product Placement		Product Placement	/productplacements
Product Geometric Representation		Product Geometric Representation	/productgeometricrepresentations
Body Geometry		Body Geometry	/bodygeometries
Body SurfaceModel Geometry		Body SurfaceModel Geometry	/bodysurfacemodelgeometries
Body SweptSolid Geometry		Body SweptSolid Geometry	/bodysweptsolidgeometries
Product Type Shape	Product Type Shape		/producttypeshapes
Type Body Geometry		Type Body Geometry	/typebodygeometries
Composition	Composition		/compositions

Object Aggregation		Object Aggregation	/objectaggregations
Element Composition		Element Composition	/elementcompositions
Spatial Composition		Spatial Composition	/spatialcompositions
Element Voiding		Element Voiding	/elementvoidings
Assignment	Assignment		/compositions
Group Assignment		Group Assignment	/groupassignments
Connectivity	Connectivity		/connectivities
Spatial Containment		Spatial Containment	/spatialcontainments
Element Filling		Element Filling	/elementfillings
Root Tracking	Root Tracking		/roottrackings
Identity		Identity	/identitys
Revision Control			/revisioncontrols
PCI	PCI		/pcis
Building Contained in Site		Building Contained in Site	/pci043
Building Storey Contained in Building		Building Storey Contained in Building	/pci044
Grid Name		Grid Name	/pci047
Grid Representation		Grid Representation	/pci048
Grid Axis Assignment		Grid Axis Assignment	/pci050
Assignment of Approval		Assignment of Approval	/pci059
Assignment of Actor		Assignment of Actor	/pci060
Absolute Placement of Building Elements		Absolute Placement of Building Elements	/pci064
Piece Mark of Building Element		Piece Mark of Building Element	/pci067
Arbitrary Profile		Arbitrary Profile	/pci069
Arbitrary Profile with Voids		Arbitrary Profile with Voids	/pci070
Pset_ PrecastDesignProperties		Pset_ PrecastDesignProperties	/pci077

Site Geometric Curve Representation	Site Geometric Curve Representation	/pci096
Precast Connection Component Assignment	Precast Connection Component Assignment	/pci136
Precast Seam Connection Reaction	Precast Seam Connection Reaction	/pci142
Precast Projection Attributes	Precast Projection Attributes	/pci145
Precast Projection Element Assignment	Precast Projection Element Assignment	/pci146
Precast Joint Attributes	Precast Joint Attributes	/pci147
Precast Joint Element Assignment	Precast Joint Element Assignment	/pci148
Precast Joint Location	Precast Joint Location	/pci149
Precast Surface Treatment Assignment	Precast Surface Treatment Assignment	/pci159
Generic Bounded Surface Geometry	Generic Bounded Surface Geometry	/pci170
Site Attributes	Site Attributes	/mvc880
Project Name	Project Name	/mvc890
Building Attributes	Building Attributes	/mvc893
Building Storey Attributes	Building Storey Attributes	/mvc895

Table 33 shows IFC concepts that are used for EMPC.1 exchange and it maps their associated REST resource collections. The URI pattern follows the structure explained in chapter 5 regarding collections and sub-collections. The URIs for individual collections are listed in Table 33. EMPC.1 includes 49 distinct REST sub-collections under 11 main collections.

6.3 Part 3: BIM Synapse Implementation

Moving to more technical aspect of implementing the framework for Cloud-BIM interoperability, this section explains what technologies are utilized to implement the framework and build the service.

6.3.1 Framework Components

In BIM data exchange in the Cloud, the interactions are between the providers of BIM service containing BIM data i.e. servers, and BIM data requesters i.e. clients. This research has proposed to use a RESTful IFC API to guide the data transfer between server and client of the BIM data. To enable the API, it should be integrated with both server and client so that the server can provide data through the API and the client can retrieve the data over HTTP request/response.

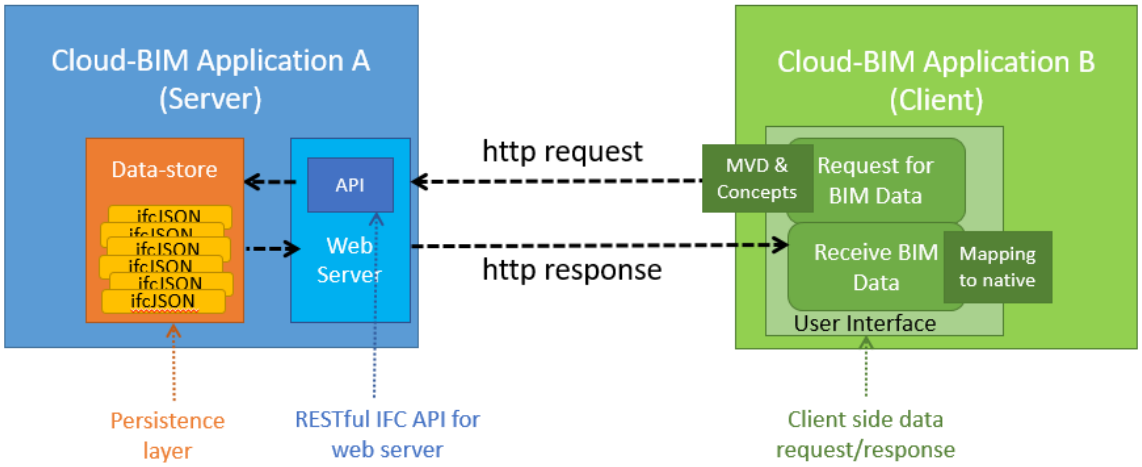


Figure 52 Three major implementation components of the framework

Figure 52 illustrates three major components of the framework implementation. These three components create three layers including a persistence layer in the server side to store ifcJSON resources for the API, the layer for REST API implemented based on IFC specification, and the integration layer in client side for interactions with the API to request and receive required data. List of technologies used for each implementation layer is shown in Figure 53 and are discussed in more detail in the following sections.

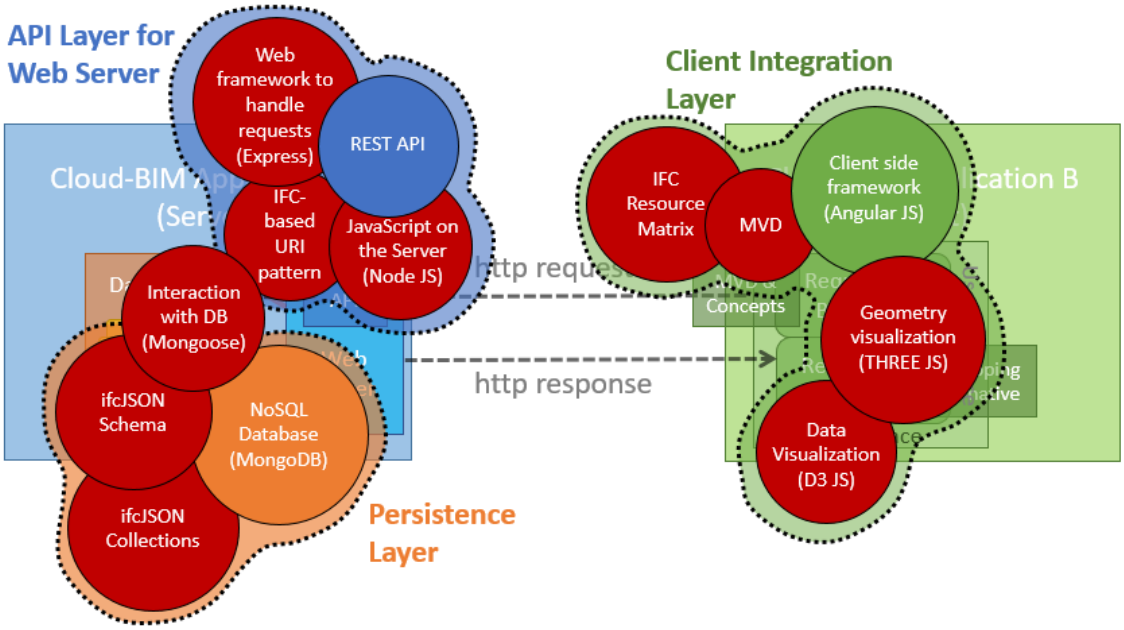


Figure 53 Overview of the technologies used in three implementation layers

Figure 54 diagrams the dataflow from the sender of the BIM data (i.e. server side) to the receiver of the BIM model (i.e. client side). In the server side, BIM data in the form of ifcJSON collections reside in the API and are maintained in the database in the form of Binary ifcJSON. In the client side, ifcJSON collections are requested and received based on a specific MVD and they are merged with the ifcJSON BIM model so that it will be

translated later to the receiving application's native bindings. The following sections will discuss each of the components with its data representation for the dataflow.

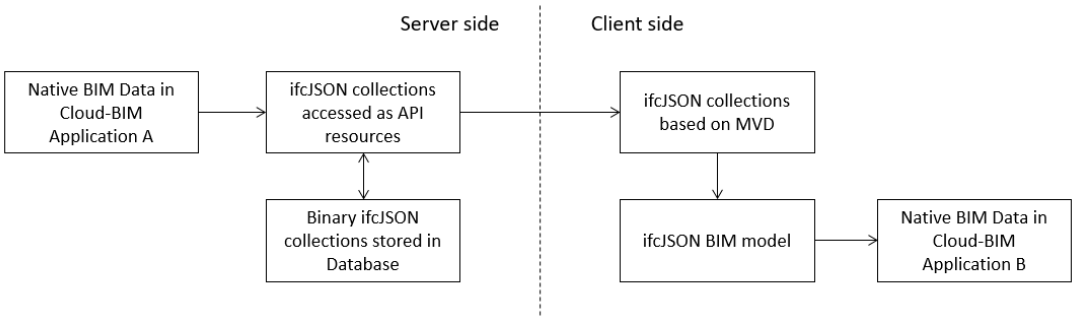


Figure 54 Dataflow from server to client native environments

6.3.2 Server Side Persistence layer

To create a repository for REST resources, this study uses MongoDB which is a NoSQL database. NoSQL which stands for "Not Only SQL" is a database mechanism that stores data different from tabular relations in relational database. NoSQL databases can horizontally scale over many servers to support a large number of operations, distribute data over many servers and allow data to be added dynamically [103, 104, 105]. Since NoSQL databases are scalable, they are becoming more popular and have been widely used [103] in the backend of major services such as eBay, SourceForge, and The Weather Channel. Instead of two-dimensional table structures, some NoSQL databases store documents and provide indexes on documents [105].

MongoDB is a scalable open-source NoSQL database that provides indexes on collections [105]. Relational database systems using SQL and MongoDB have been compared in the work of [103]. They have pointed out that MongoDB has better runtime performance than SQL for inserts, updates and simple queries. MongoDB could be a good

solution for larger data sets with changing schema. unlike SQL that requires a well-defined and rigid schema, MongoDB can work with a dynamic schema. Even in working with structured data with strict schema, MongoDB generally performs better than SQL [103]. In MongoDB, collections of documents are indexed and the queries are performed on documents. MongoDB stores data in BSON format, a binary JSON-like format. Its client driver serializes the documents into BSON and then stores them on MongoDB server [105]. Figure 55 illustrates major differences between Relational Database and MongoDB.

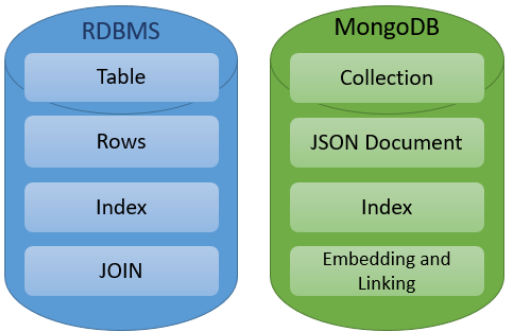


Figure 55 Differences in database mechanism between Relational Database Management System (RDBMS) and MongoDB

MongoDB is used in the implementation of the persistence layer of the framework to persist the API's data. REST collections discussed in Chapter 5 are encoded in ifcJSON and maintained as MongoDB collections directly. MongoDB serializes the ifcJSON documents in the backend to store them on its server.

Table 34 lists the top-level collections and sub-collections used in the database implemented in MongoDB that follows IFC specification and MVD concepts. These collections correspond to REST resources. Users can access the data by its ID within the collections.

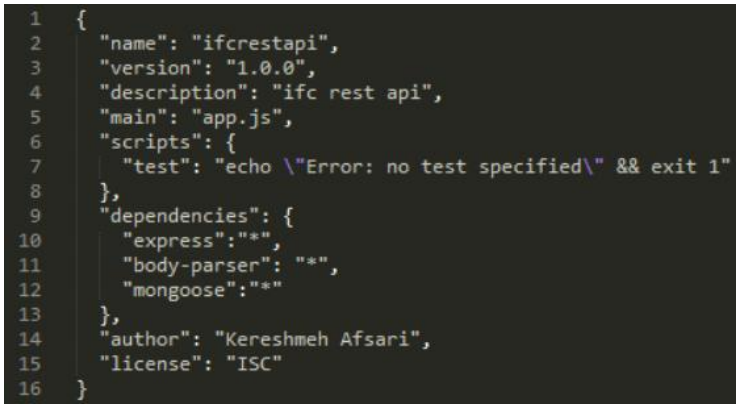
Table 34 Top-level database collections and sub-collections

Database Main Collections	Corresponding IFC concept	Sub-Collections
objectdefinitions	Object Definition	objectdefinitions, objecttypings
roottrackings	Root Tracking	revisioncontrols, identities
productshapes	Product Shape	productgeometricrepresentations, producttopologyrepresentations, productplacements
propertysets	Property Sets	propertysetsforobjects, propertysetsfortypes, quantitysets
producttypeshapes	Product Type Shape	typeaxisgeometries, typebodygeometries, typelightinggeometries
resources	Resource	resourcecosts, resourcequantities
projects	Project	projectdeclarations, projectunitss, projectcontexts, projectclassificationinformations, projectdocumentinformations, projectlibraryinformations
compositions	Composition	objectaggregations, elementvoidings, objectnestings, portnestings, typebasedports
objectassociations	Object Association	objectclassifications, objectdocumentations, objectlibrarys, objectapprovals, objectconstraints, materials
assignments	Assignment	actorassignments, controlassignments, groupassignments, productassignments, processassignments, resourceassignments, producttypeassignments, processtypeassignments, resourcetypeassignments
connectivities	Connectivity	spatialstructures, spaceboundaries, elementconnectivities, controlflows, elementfilings, structuralactivities, structuralconnectivities, sequentialconnectivities
pcis	PCI	pci043, pci044, pci047, pci048, pci050, pci059, pci060, pci064, pci067, pci069, pci070, pci077, pci096, pci136, pci142, pci145, pci146, pci147, pci148, pci149, pci159, pci170, mvc880, mvc890, mvc893, mvc895

6.3.3 IFC REST API for the Web Server

IFC REST API on the server side acts as an interface for querying and persisting data in the MongoDB database. This API for the web server in the sending application enables the collaborating Cloud-BIM application (i.e. client for this server) to request and receive the required BIM data.

Node.js is an open-source JavaScript runtime environment that is built on Google Chrome's engine and enables executing scalable server applications. It is basically JavaScript on the server. Also, Express.js is an open source web application framework for Node.js that helps to organize the software architecture and handle the requests on the server side. Here, the Node.js and Express.js run on localhost port 3000. The gateway to the application is app.js and running “npm init” locally in the server folder will create the package.json which manages the application dependencies as shown in Figure 56. Then, running “npm install” will install the dependencies for the API. In app.js the dependencies that are needed should be included within the required variables as shown in Figure 57. Figure 57 also shows how the base URL for IFC REST API is set.



```
{
  "name": "ifcrestapi",
  "version": "1.0.0",
  "description": "ifc rest api",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "dependencies": {
    "express":"*",
    "body-parser": "*",
    "mongoose":"*"
  },
  "author": "Kereshmeh Afsari",
  "license": "ISC"
}
```

Figure 56 package.json for IFC REST API

```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');
var mongoose = require('mongoose');

app.use(express.static(__dirname+ '/client'));

// Connect to Mongoose
mongoose.connect('mongodb://localhost/ifcrestapi');
var db = mongoose.connection;

app.get('/', function(req, res){
    res.send('IFC REST API');
});


app.listen(3000);
console.log('Running on port 3000');
```

Figure 57 Required variables, dependencies, and connections in app.js

Mongoose is used to let the REST API interact with MongoDB database. Mongoose is an open source Object Data Modeling (ODM) library that provides a modeling environment for the data and wraps the Node.js driver. Mongoose allows to define and enforce schemas for the collections.

```
var mongoose = require('mongoose');

// Placement Schema
var placementSchema = mongoose.Schema({
    instanceId: {
        type: Number,
        required: true
    },
    placementRelTo: {
        type: Object,
        required: true
    },
    relativePlacement: {
        type: Object,
        required: true
    }
});

var Placement = module.exports = mongoose.model('Placement', placementSchema);

// Get Placements
module.exports.getPlacements = function(callback, limit){
    Placement.find(callback).limit(limit);
}
```

Figure 58 productplacement object model with Mongoose

Figure 58 shows how to establish a connection and object model with Mongoose for a productplacement object. It defines the schema for object placement based on IFC concept “Product Placement”. Figure 59 shows the ifcbuildingelement object model with

Mongoose that follows the IFC "Object Definition" concept explained previously in Figure 39. Mongoose schema for each object model follows the ifcJSON schema for each collection and each collection schema corresponds to either the associated IFC fundamental concepts or MVD concepts.

```
var mongoose = require('mongoose');

// buildingelement Schema
var buildingelementSchema = mongoose.Schema({
    instanceId: {type: Number, required: true },
    globalId: {type: String, required: true },
    ownerHistory: {type: Object, required: true },
    name: {type: String, required: true },
    description: {type: String, required: true },
    objectType: {type: String, required: true},
    objectPlacement: {type: Object, required: true},
    representation: {type: Object, required: true},
    tag: {type: String, required: true },
    predefinedType: {type: String, required: true},
});

var buildingelement = module.exports = mongoose.model('buildingelement', buildingelementSchema);

// Get buildingelements
module.exports.getBuildingelements = function(callback, limit){
    Buildingelement.find(callback).limit(limit);
}

// Get buildingelement
module.exports.getBuildingelementById = function(id, callback){
    Buildingelement.findById(id, callback);
}
```

Figure 59 ifcbuildingelement object model with Mongoose

REST APIs are designed for consumers and the structure of URIs should have meanings for consumers too. Although it is not required to use a specific structure for URI patterns, it would be best to follow an understandable pattern to improve the accessibility of resources. It is usually not practical to list individual URIs but if the URIs can match a well-structured pattern, the API will become more useful. Also, URI patterns should be easily understandable by human users and should not be ambiguous or inaccurate. The IFC REST API implemented in this study, runs on localhost and the base URL to access the API is http://localhost:3000/ifsrestapi/. Table 35 lists main URIs for the API resources. As

explained in chapter 5, the URI pattern follows the IFC concepts since the API resources are also organized and designed based on IFC specification.

Table 35 URI pattern to access main ifcJSON collections

ifcJSON Main Collections	Corresponding IFC concept	URI
objectdefinitions	Object Definition	/objectdefinition
roottrackings	Root Tracking	/roottracking
productshapes	Product Shape	/productshapes
propertysets	Property Sets	/propertysets
producttypeshapes	Product Type Shape	/producttypeshapes
resources	Resource	/resources
projects	Project	/projects
compositions	Composition	/compositions
objectassociations	Object Association	/objectassociations
assignments	Assignment	/assignments
connectivities	Connectivity	/connectivities

6.3.4 Client Side Interactions

The techniques of data integration in the client side and mapping guidelines for visualization of received BIM data in the client side is not in the scope of this research since the research is intended to use the proposed framework to enable sending and receiving of complete BIM model which is in the form of web compatible data format. However, to interact with the IFC REST API and to provide a better understanding of the proposed framework, a frontend application has been created which can also be used in the client side. In the client-side Cloud-BIM application (i.e. the receiver of BIM data), an integration layer should be implemented where the receiver of BIM data can connect to IFC REST API of the server Cloud-BIM application (i.e. sender of the BIM data) and

request for the API resources to receive the ifcJSON data. Upon receiving the required BIM data, the receiving Cloud-BIM application should translate the ifcJSON BIM model to its native binding.

For the implementation of frontend application to interact with REST resources, this study uses AngularJS which is a client side framework created by Google and built on JavaScript. Angular is based on Model View Controller (MVC) architecture separating views from the application logic that consists of three parts: model for marinating data, view for displaying data, and controller that controls the interaction between model and view. Figure 60 shows an example of a controller for ifcbuildingelements that receives resources from ifcbuildingelements collection.

```
var myApp = angular.module('myApp');

myApp.controller('ifcBuildingelementsController', ['$scope', '$http', '$location', '$routeParams',
    function($scope, $http, $location, $routeParams){
    //console.log('BuildingelementsController loaded...');

    $scope.getBuildingelements = function(){
        $http.get('/api/ifcbuildingelements').success(function(response){
            $scope.buildingelements = response;
        });
    }

    $scope.getBuildingelement = function(){
        var id = $routeParams.id;
        $http.get('/api/ifcbuildingelements/'+id).success(function(response){
            $scope.buildingelement = response;
        });
    }

    function Main($scope) {
      $scope.rootFolders = '/ifcbuildingelements';
    }

}]);

myApp.controller('TabController', function (setTab){
    this.tab = 1;
    this.selectTab = function(){
        this.tab = setTab;
    };
    this.isSelected = function(checkTab){
        return this.tab === checkTab;
    };
});
```

Figure 60 Controller design for ifcbuildingelements

The view is the presentation of data when the controller triggers that. Figure 61 shows the model for ifcbuildingelements which is responsible for managing the

application data. The model is attached to $scope (shown in Figure 60) in controller and $scope can be accessed by the view to be shown in the browser in two-way data binding.

```
var MyApp = angular.module('myApp', ['ngRoute', 'directives.modelViewer']);

MyApp.config(function($routeProvider){
    $routeProvider.when('/ifcbuildingelements', {
        controller: 'BuildingelementsController',
        templateUrl: 'views/buildingelements.html'
    })
    .when('/ifcbuildingelements/details/:id', {
        controller: 'BuildingelementsController',
        templateUrl: 'views/buildingelement_detail.html'
    })
    .otherwise({
        redirectTo: '/'
    });
});

MyApp.controller('myAppController', ['$scope', '$route', function($scope, $route) {

    $scope.reloadRoute = function() {
        $route.reload();
    }

}]);
```

Figure 61 Angular model for ifcbuildingelements

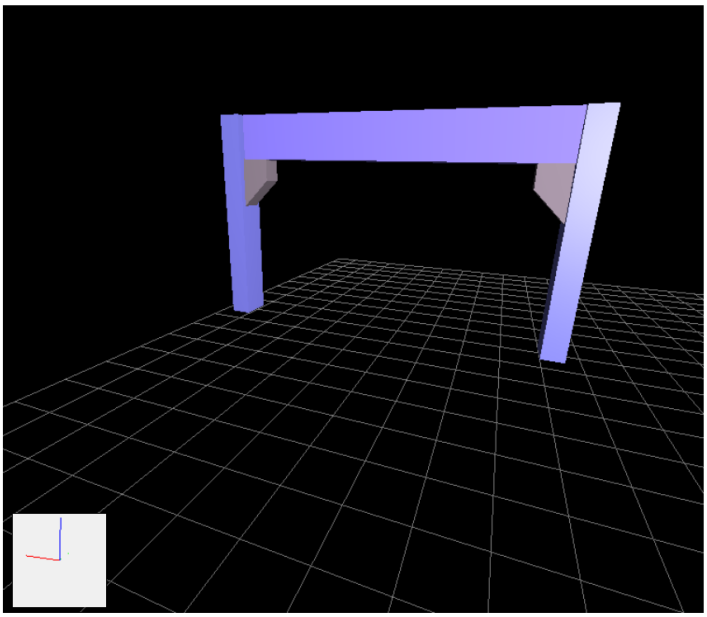


Figure 62 Visualization of ifcJSON model using Three.js

To visualize the ifcJSON in browser Three.js is used which is an open source JavaScript library with a WebGL renderer for drawing 3D graphics on the web. Three.js

in the frontend visualizes the ifcJSON BIM model which includes BIM objects based on the retrieved data. Figure 62 shows the implemented Three.js interface in the browser. Mapping ifcJSON resources to Three.js native geometry is an example of direct and easy mapping between IFC REST API resources and native bindings. But as mentioned data mapping and integration in the client side is not in the scope of this research.

To retrieve data from a server REST API, as mentioned in chapter 5, a client only needs to know the location of the resources. The location of the resources i.e. collection are specified by URIs. URIs are designed based on IFC fundamental concepts and MVD-specific concepts. Each MVD contains a set of concepts and requesting for a BIM model based on an MVD set of concepts will return ifcJSON resources that are included in the MVD. Figure 63 illustrates the client-side interface that provides the interaction with IFC REST API. The "URL" section allows the user to input the URL of the base IFC REST API of the sending Cloud-BIM application to connect to that. In this study, since the server is running on localhost the base URL will be http://localhost:300/ifcrestapi. The HTTP actions are listed under "Method" which in this study as mentioned will only look at GET operation since the study is dealing with data retrieval. The "MVD" drop down menu, lists all the MVDs that the BIM data can be retrieved upon.

Under "Resources", all URIs that are related to API collections including IFC concepts and MVD-specific concepts are listed. In the top level, it lists the URIs that are previously shown in Table 35. The "Show All" option will retrieve all resources within the MVD while selecting a specific resource will retrieve only that collection in addition to the resources that are linked to that collection.

Figure 63 Client side interaction with IFC REST API resources based on MVD and IFC concepts

6.3.5 MVD and REST Resources

In this study, MVD for precast concrete has been revised as explained in chapter 6 to both upgrade to IFC4 specification and integrate IFC4 fundamental concepts with MVD specific concepts. This revised MVD has been implemented in ifcDoc for validation purposes.

Within 49 concepts identified in EMPC.1 Exchange model, 23 concepts belong to IFC4 specification and 26 concepts are revised PCI-specific concepts. For BIM Synapse implementation in this study, 20 REST resource collections among 49 concepts are used as summarized in Figure 64. These REST resources correspond to 11 IFC4 fundamental concepts and 9 PCI MVD concepts. REST resources that correspond to IFC 4 concepts address 7 top-level collections including Project, Object Definition, Property Sets, Product Shape, Composition, Connectivity, and Root Tracking. These selected REST resources include precast elements with their geometric representation in the BIM model with required aggregation both in the element level and project's spatial hierarchy. They also carry PCI specific data regarding projection elements and spatial elements requirements. Data regarding precast object association, element type, group assignment, joint and

surface treatment are excluded. The list of URIs that correspond to these REST resources is represented in Table 36.

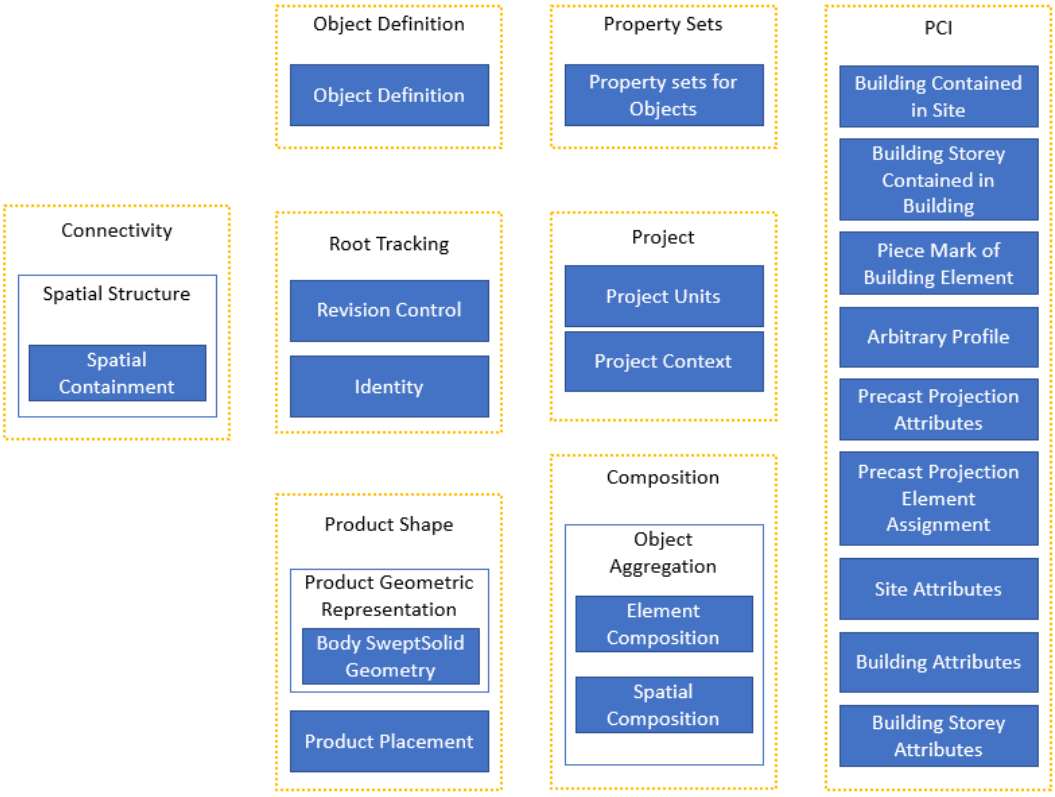


Figure 64 REST resource collections selected for implementation

In this implementation, some of the collections are narrowed down to their low-level sub-collections: Product Geometric Representation collection with its surface model geometry is narrowed down to Body SweptSolid Geometry sub-collection, Object Aggregation collection is narrowed down to two sub-collections including Element Composition and Spatial Composition, and Connectivity collection is narrowed down to Spatial Containment sub-collection. REST resources that correspond to PCI MVD include Building Contained in Site (PCI-043), Building Storey Contained in Building (PCI-044), Piece Mark of Building Element (PCI-067), Arbitrary Profile (PCI-069), Precast Projection

Attributes (PCI-145), Precast Projection Element Assignment (PCI-146), Site Attributes (MVC-880), Building Attributes (MVC-893), Building Storey Attributes (MVC-895). Figure 65 shows the concept matrix for EMPC.1 that is implemented in IfcDoc tool.

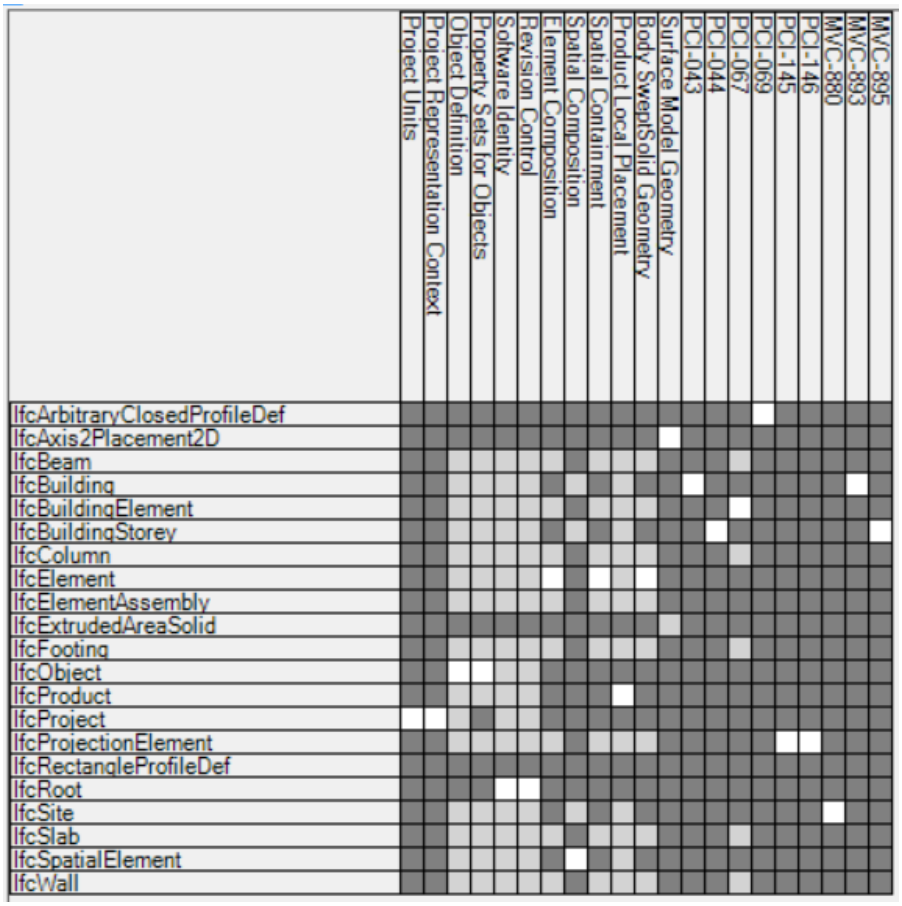


Figure 65 Concept matrix implementation for EMPC.1 in IfcDoc tool (White: the entity is directly involved in the validation of the corresponding concept, Light Grey: the entity is involved in the validation of the corresponding concept through its superclass, Dark Grey: the entity is not involved in the validation of the corresponding concept)

Table 36 List of URIs for top-level REST collections

URIs
/ifcrestapi/objectdefinitions/ifcproducts/ifcelements/ifcbuildingelements
/ifcrestapi/objectdefinitions/ifcproducts/ifcelements/ifcelementassemblies
/ifcrestapi/objectdefinitions/ifcproducts/ifcelements/ifcfeatureelements
/ifcrestapi/objectdefinitions/ifcproducts/ifcelements/ifcspatialstrutureelements
/ifcrestapi/productshapes/productplacements
/ifcrestapi/productshapes/bodysweptsolidgeometries
/ifcrestapi/roottrackings/revisioncontrols
/ifcrestapi/roottrackings/identities
/ifcrestapi/connectivities/spatialcontainments
/ifcrestapi/propertysets/propertysetsforobjects
/ifcrestapi/projects/projectunits
/ifcrestapi/projects/projectcontexts
/ifcrestapi/compositions/objectaggregations/elementcompositions
/ifcrestapi/compositions/objectaggregations/spatialcompositions
/ifcrestapi/pcis/buildingcontainedinsites
/ifcrestapi/pcis/buildingstoreycontainedinsites
/ifcrestapi/pcis/piecemarkofbuildingelements
/ifcrestapi/pcis/arbitratyprofiles
/ifcrestapi/pcis/precastprojectionattributes
/ifcrestapi/pcis/precastprojectionelementassignments
/ifcrestapi/pcis/siteattributes
/ifcrestapi/pcis/buildingattributes
/ifcrestapi/pcis/buildingstoreyattributes

6.4 Part 4: Test Cases and Workflow

To evaluate the framework, test cases have been developed. In EMPC.1 exchange, model contains data for conceptual design of precast concrete pieces to address requirements of the exchange. The model must conform to PCI MVD and its 20 concepts that are implemented here. To illustrate the BIM data, a test scenario is shown in Figure 66. This BIM model which is an architectural model of a precast building is part of a precast concrete parking deck that contains major precast pieces such as column, beam, slab, wall, and foundation. The model has 4 levels including the basement level for the foundations.

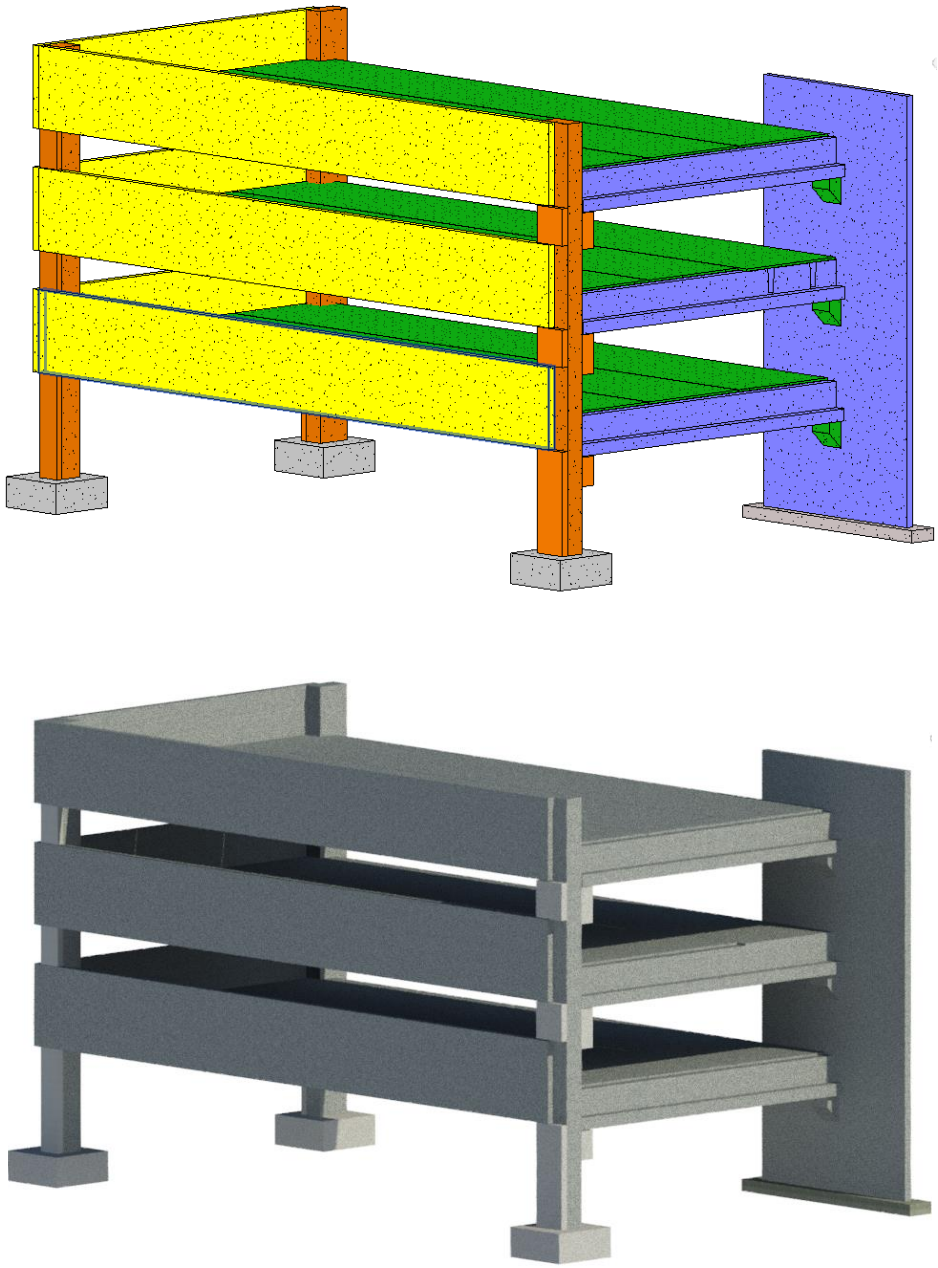


Figure 66 Precast Concrete BIM model as a test case

This model is a stand-alone structure designed with standard precast concrete elements with a level of detail appropriate to conceptual phase i.e. LOD 200. The precast concrete components in this BIM model are listed in Table 37.

Table 37 Components of the BIM model in the test case

Main Components	Sub-elements	Count
Building		1
Site		1
Levels		4
Precast Columns with Projection Elements	Columns	3
	Corbels	15
Precast Inverted Tee Beams		3
Precast Rectangular Beams		6
Precast Slabs Assemblies with Double Tees	Slab Assemblies	3
	Double Tees in each Slab	3
Precast Foundations		3
Precast Footing		1
Precast Wall with Projection Elements	Wall	1
	Corbels	3

The process of generating the test case for evaluation is illustrated in Figure 67. The process begins by generating an IFC model that conforms to both IFC4 schema and PCI MVD to ensure that the IFC model is correct and complete. Then, the model is translated to ifcJSON and modularized based on REST collections described earlier to be stored in the database and perform as REST resources.

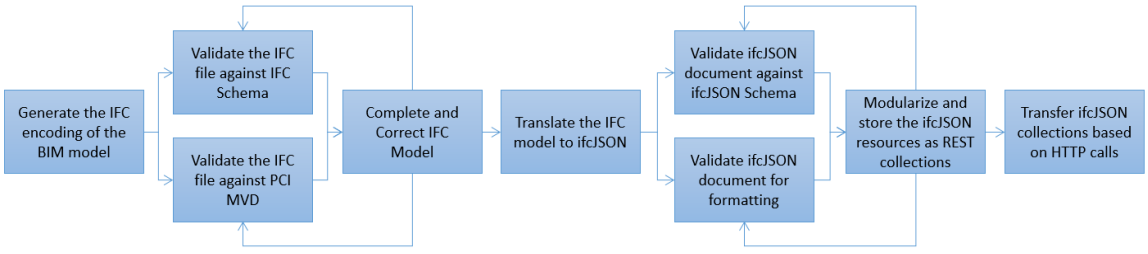


Figure 67 The process of test case generation

6.4.1 IFC Model

The test model is translated to IFC using both automated translation and manual coding to ensure correct mapping of IFC entities. The BIM model visualized in Solibri Model Viewer is shown in Figure 68.

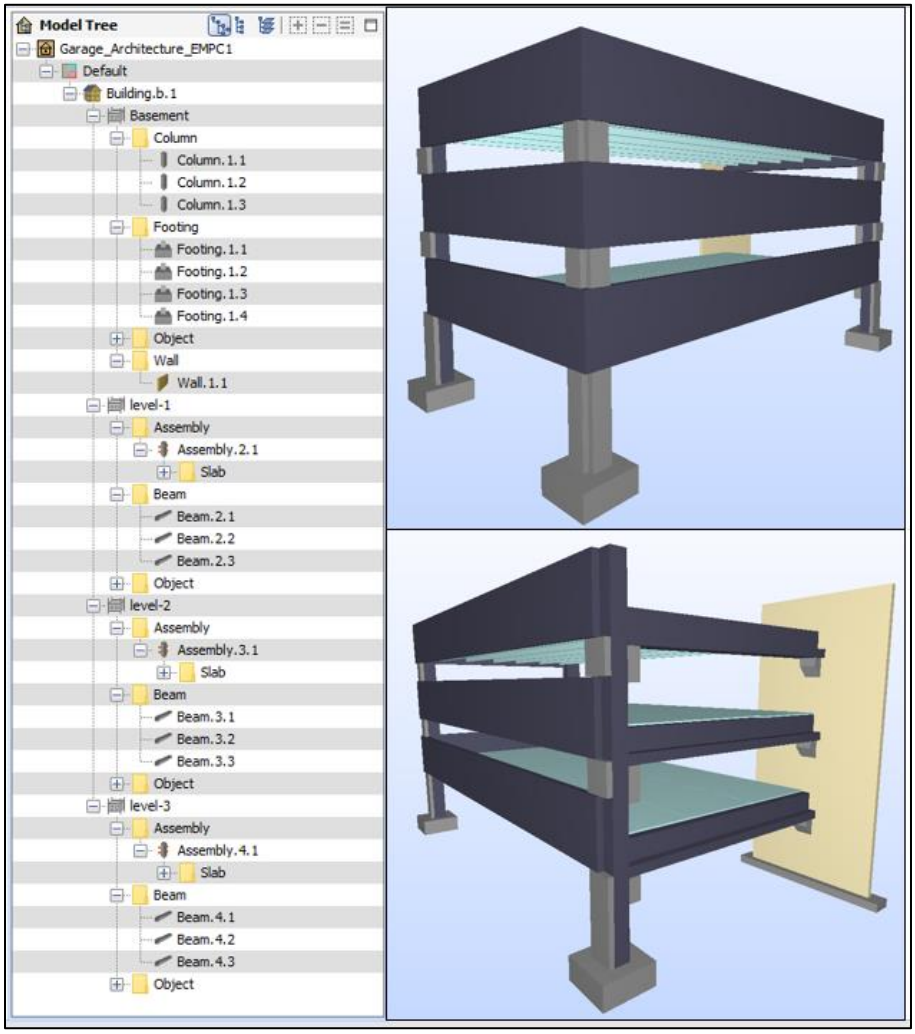


Figure 68 IFC model and its component hierarchy

In this IFC model there are a total number of 83 IFC entities in IfcRoot level and 53 IfcObject entities. The number of different IFC entities in the model is listed in Table 38. Figure 69 shows major elements in the IFC model including column, beams, and wall.

Table 38 Main IFC entities in the BIM model

IFC Entities				Count
IfcProject				1
IfcSite				1
IfcBuilding				1
IfcBuildingStorey				4
IfcExtrudedAreaSolid				42
IfcProduct				53
	IfcSpatialElement			6
	IfcElement			47
		IfcElementAssembly		3
		IfcProjectionElement		18
		IfcBuildingElement		26
			IfcBeam	9
			IfcColumn	3
			IfcFooting	4
			IfcSlab	9
			IfcWall	1

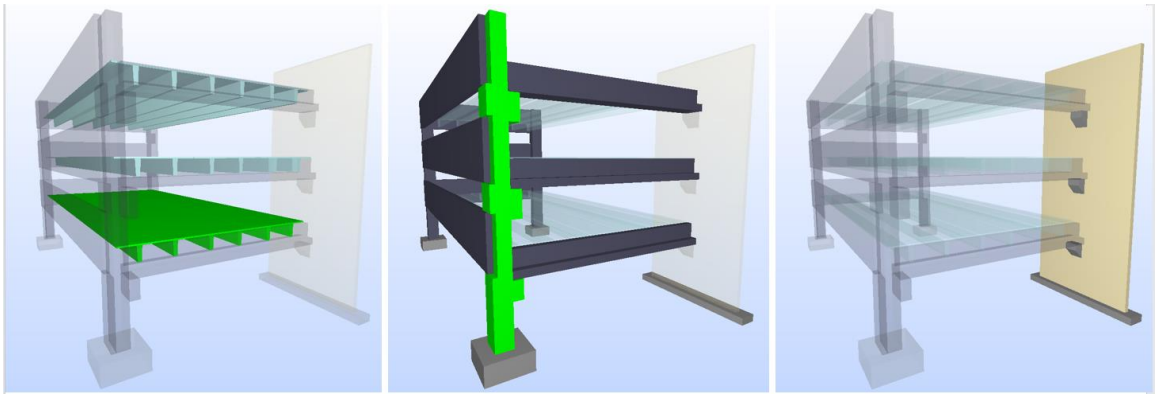


Figure 69 Major elements in IFC model

In each floor of the model, i.e. level 1, 2, and 3 there is a base slab as an assembly of three IfcSlab entities which represent the double tees. These IfcSlab instances are aggregated through an IfcElementAssembly and IfcRelAggregates entity. Figure 70 shows one of the double tees and the aggregation of the slab in the IFC model.

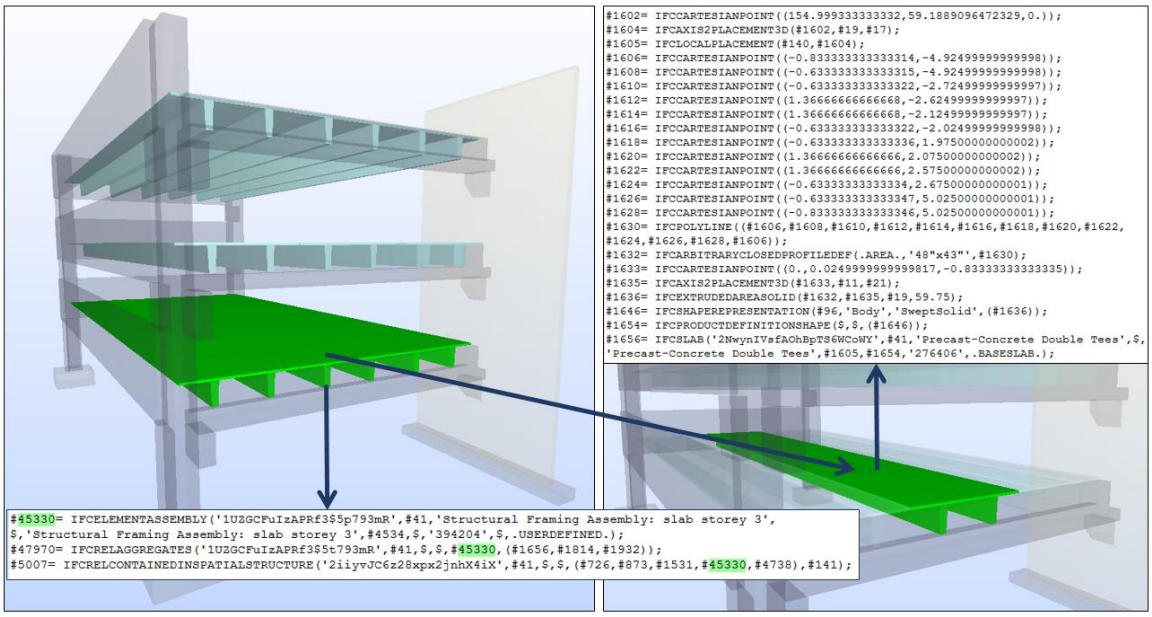


Figure 70 Level 1 double tee slab in the IFC model

Figure 71 Inverted tee beam and rectangular beam in level 2 of the IFC model

In addition, in level 1, 2, and 3 there are three beam instances each represented by IfcBeam entity as shown in Figure 71. In each floor, one of the beams is an inverted tee beam and two others are rectangular beams. The inverted tee beam performs as girders receiving double tees and is supported by two corbels: the corbel in the wall and the corbel in the column. The "PredefinedType" attribute of all IfcBeam entities is considered as "BEAM" because in IFC4 specification, the IfcBeamTypeEnum does not include any predefined type for inverted tee beams.

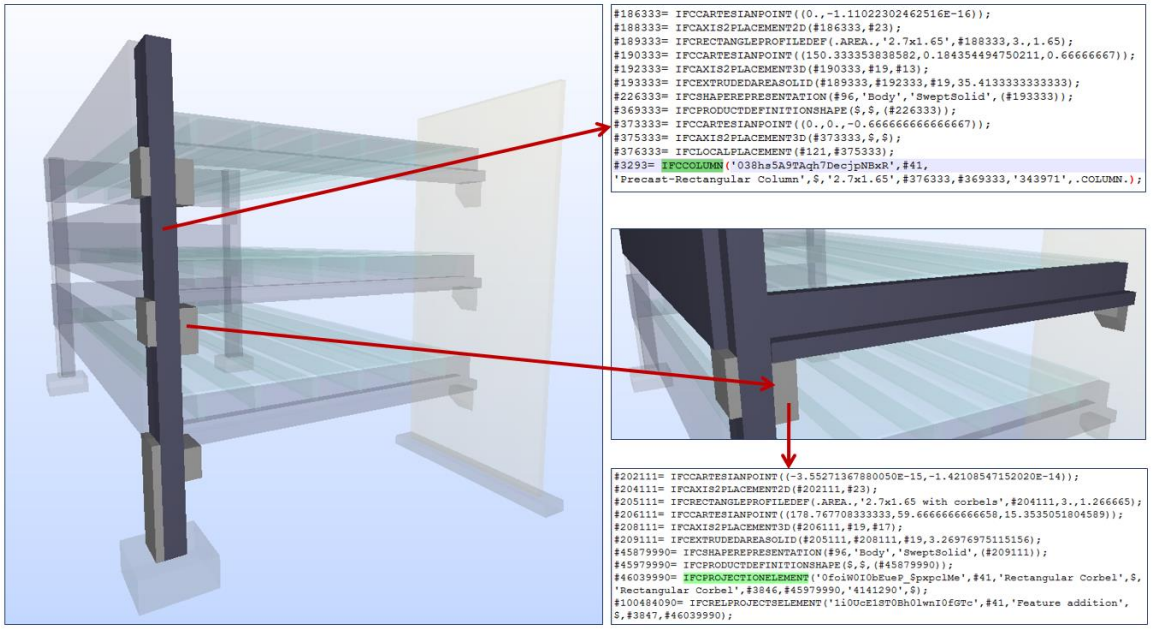


Figure 72 Column and corbels in the IFC model

There are three columns in the model and each have multiple corbels projecting from the faces of the precast concrete columns to support primary beams. Corbels in this IFC model, as shown in Figure 72, are mapped to IfcProjectionElement since PCI MVD requires projections to be presented by IfcProjectionElement. An IfcRelProjectsElement creates a relationship between the column and its corbels. Since in IFC4 schema, the

IfcRelProjectsElement relationship is defined as a 1:1 relationship, there is one IfcRelProjectsElement for each projection.

After generating the IFC model, the IFC model needs to be validated both against the IFC schema for syntax checking and against the PCI MVD for exchange requirement validation. For checking against IFC4 schema, EXPRESS-O engine is used (shown in Figure 73). Since the study needs to validate the model against the MVD requirements, the PCI MVD is revised which is explained earlier in the "Application" section. Revised PCI MVD is implemented for EMPC.1 in IfcDoc tool that contains the 20 selected concepts. Since in EMPC.1 all concepts are considered as "required" modules, concepts are assigned as mandatory in the implementation of MVD in IfcDoc tool. The HTML result of validation against MVD is shown in Figure 74.

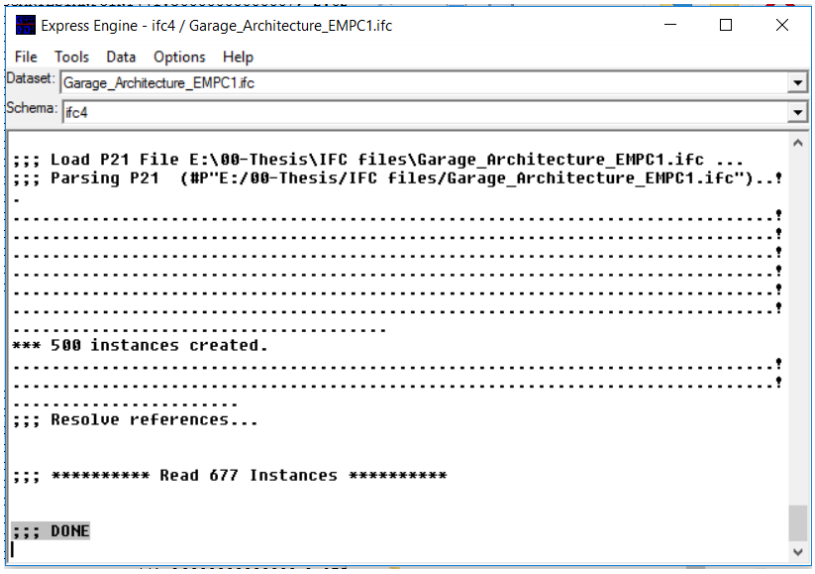


Figure 73 Validation of the test model against IFC schema

Validation Results

Instance File	E:\00-Thesis\IFC files\Garage_Architecture_EMPC1.ifc
Project File	E:\00-Thesis\ifcDoc\Revised-PCI_forREST.ifcdoc
Model View	PCI MVD
Exchange	EMPC1
Tests Executed	20
Tests Passed	20
Tests Ignored	0
Tests Percentage	100%

Figure 74 Model validation result in HTML format

Figure 75 shows the model validation in IfcDoc tool with associated concepts and IFC entities that are tested in the validation. The matrix definition was already explained in Figure 65.

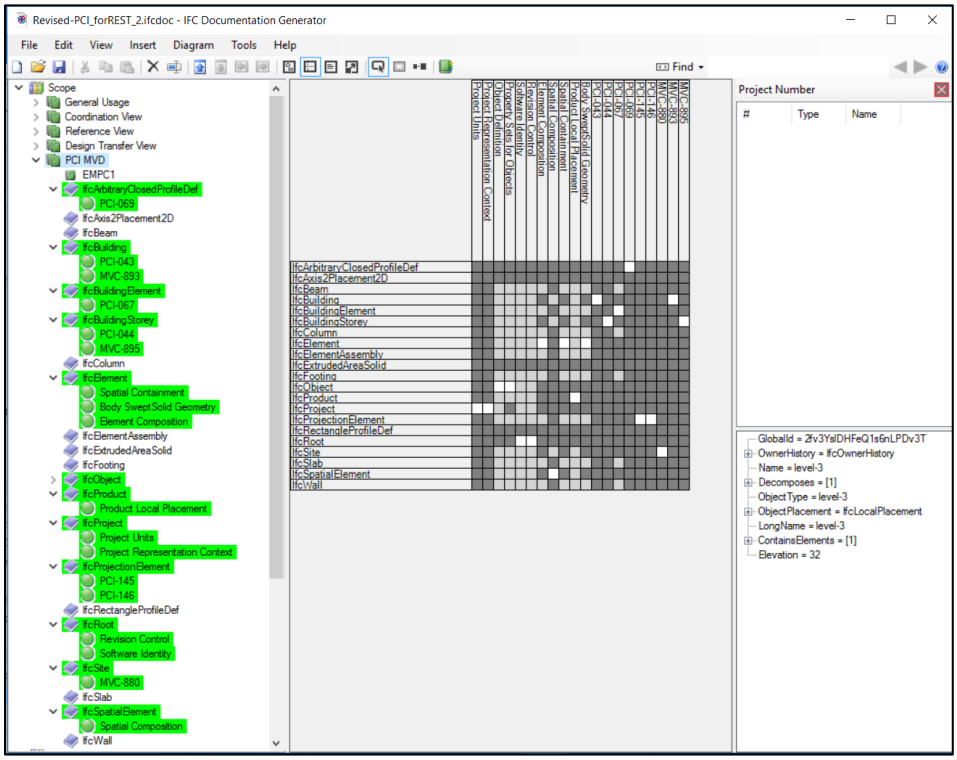
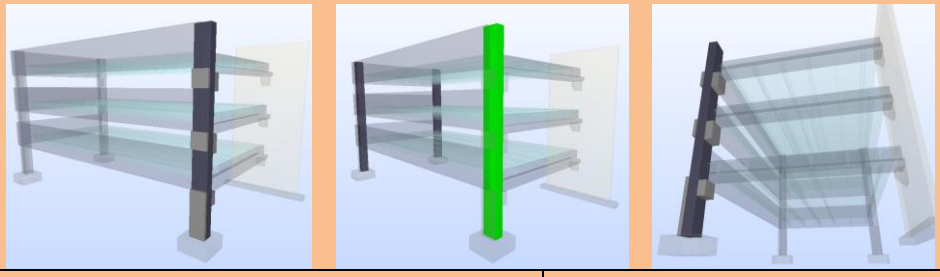
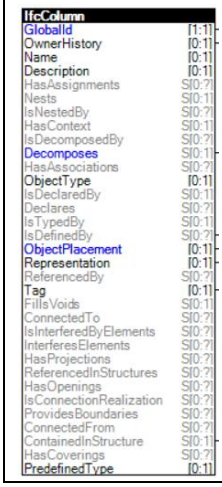


Figure 75 Model validation results in IfcDoc tool- Green shows the test passes an item and all items within

6.4.2 ifcJSON REST Resources

To transform the test case to REST resources the BIM model needs to be serialized in ifcJSON and mapped to REST resources. Table 39 shows the mapping guide for the precast column to REST resources. The URIs are narrowed down to sub-collections for example for the geometry representation Body SweptSolid geometry is used directly.

Table 39 Mapping guide for precast concrete column

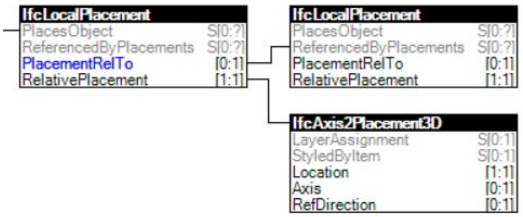
<table>
<tr><td colspan="2">Schema: IFC4 Add2</td></tr>
<tr><td colspan="2">MVD: Precast Concrete</td></tr>
<tr><td colspan="2">Exchange Model: EMPC.1</td></tr>
<tr><td colspan="2">Data: Precast Concrete Column
  </td></tr>
<tr><td>IFC SPF Data and Diagram</td><td>ifcJSON REST Resources</td></tr>
<tr><td colspan="2">Object Definition (ifcbuildingelement)</td></tr>
<tr><td colspan="2">URI: <code>/ifcrestapi/objectdefinitions/ifcproducts/ifcelements/ifcbuildingelements</code></td></tr>
<tr><td>
<pre>#3293=
IFCCOLUMN('038hs5A9TAqh7DecjpNBxR',
#41,'Precast-Rectangular Column',
$,'2.7x1.65',#376333,#369333,'343971',
.COLUMN.);</pre>


</td><td>
<pre>{
  "instanceId": 3293,
  "ifc": "IFCCOLUMN",
  "globalId": "038hs5A9TAqh7DecjpNBxR",
  "ownerHistory": {
    "instanceId": 41
  },
  "name": "Precast-Rectangular Column",
  "description": null,
  "objectType": "2.7x1.65",
  "objectPlacement": {
    "instanceId": 376333
  },
  "representation": {
    "instanceId": 369333
  },
  "tag": "343971",
  "predefinedType": "COLUMN"
}</pre>
</td></tr>
</table>

Product Placement

URI: `/ifcrestapi/productshapes/productplacements`

```
#6= IFCCARTESIANPOINT((0.,0.,0.));
#4813= IFCAXIS2PLACEMENT3D(#6,$,$);
#4814= IFCLOCALPLACEMENT($,#4813);
#31= IFCAXIS2PLACEMENT3D(#6,$,$);
#32= IFCLOCALPLACEMENT(#4814,#31);
#118= IFCCARTESIANPOINT
((0.,0.,0.666666666666667));
#120= IFCAXIS2PLACEMENT3D(#118,$,$);
#121= IFCLOCALPLACEMENT(#32,#120);
#373333= IFCCARTESIANPOINT
((0.,0.,-0.666666666666667));
#375333= IFCAXIS2PLACEMENT3D(#373333,$,$);
#376333= IFCLOCALPLACEMENT(#121,#375333);
```

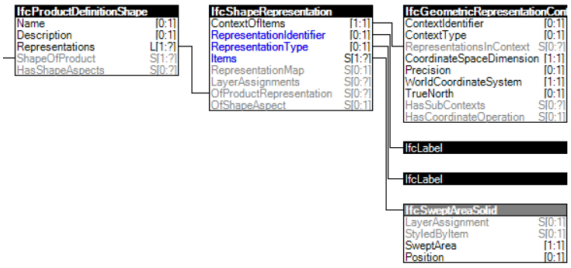



```
{
  "instanceId": 376333,
  "ifc": "IFCLOCALPLACEMENT",
  "placementRelTo": {
    "instanceId": 121
  },
  "relativePlacement": {
    "instanceId": 375333,
    "ifc": "IFCAXIS2PLACEMENT3D",
    "location": {
      "instanceId": 373333,
      "ifc": "IFCCARTESIANPOINT",
      "coordinates": [0, 0, -0.666666666666667]
    },
    "axis": null,
    "refDirection": null
  }
},
{
  "instanceId": 121,
  "ifc": "IFCLOCALPLACEMENT",
  "placementRelTo": {
    "instanceId": 32
  },
  "relativePlacement": {
    "instanceId": 120,
    "ifc": "IFCAXIS2PLACEMENT3D",
    "location": {
      "instanceId": 118,
      "ifc": "IFCCARTESIANPOINT",
      "coordinates": [0, 0, 0.666666666666667]
    },
    "axis": null,
    "refDirection": null
  }
},
{
  "instanceId": 32,
  "ifc": "IFCLOCALPLACEMENT",
  "placementRelTo": {
    "instanceId": 4814
  },
  "relativePlacement": {
    "instanceId": 31,
    "ifc": "IFCAXIS2PLACEMENT3D",
    "location": {
      "instanceId": 6,
      "ifc": "IFCCARTESIANPOINT",
      "coordinates": [0, 0, 0]
    },
    "axis": null,
    "refDirection": null
  }
},
{
  "instanceId": 4814,
  "ifc": "IFCLOCALPLACEMENT",
  "placementRelTo": null,
  "relativePlacement": {
    "instanceId": 4813,
    "ifc": "IFCAXIS2PLACEMENT3D",
    "location": {
      "instanceId": 6,
      "ifc": "IFCCARTESIANPOINT",
      "coordinates": [0, 0, 0]
    },
    "axis": null,
    "refDirection": null
  }
}
```

Body SweptSolid Geometry

URI: `/ifcrestapi/productshapes/bodysweptsolidgeometries`

```
#13= IFCDIRECTION((-1.,0.,0.));
#19= IFCDIRECTION((0.,0.,1.));
#23= IFCDIRECTION((1.,0.));
#186333= IFCCARTESIANPOINT((0.,0.));
#188333= IFCAXIS2PLACEMENT2D(#186333,#23);
#189333= IFCRECTANGLEPROFILEDEF
(.AREA.,'2.7x1.65',#188333,3.,1.65);
#190333= IFCCARTESIANPOINT
((150.333353838582,0.184354494750211,0.666666666666667));
#192333= IFCAXIS2PLACEMENT3D(#190333,#19,#13);
#193333= IFCEXTRUDEDAREASOLID
(#189333,#192333,#19,35.4133333333333);
#226333= IFCSHAPEREPRESENTATION
(#96,'Body','SweptSolid',(#193333));
#369333= IFCPRODUCTDEFINITIONSHAPE($,$,(#226333));
```

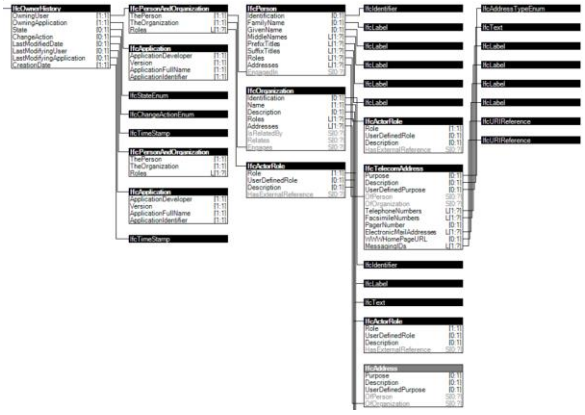



```
{
  "instanceId": 369333,
  "ifc": "IFCPRODUCTDEFINITIONSHAPE",
  "name": null,
  "description": null,
  "representations": [
    {
      "instanceId": 226333,
      "ifc": "IFCSHAPEREPRESENTATION",
      "contextOfItems": {
        "instanceId": 96
      },
      "representationIdentifier": "Body",
      "representationType": "SweptSolid",
      "items": [
        {
          "instanceId": 193333,
          "ifc": "IFCEXTRUDEDAREASOLID",
          "sweptArea": {
            "instanceId": 189333,
            "ifc": "IFCRECTANGLEPROFILEDEF",
            "profileType": "AREA",
            "profileName": "2.7x1.65",
            "position": {
              "instanceId": 188333,
              "ifc": "IFCAXIS2PLACEMENT2D",
              "location": {
                "instanceId": 186333,
                "ifc": "IFCCARTESIANPOINT",
                "coordinates": [
                  0,
                  0
                ]
              },
              "refDirection": {
                "instanceId": 23,
                "ifc": "IFCDIRECTION",
                "directionRatios": [
                  1,
                  0
                ]
              }
            },
            "xDim": 3,
            "yDim": 1.65
          },
          "position": {
            "instanceId": 192333,
            "ifc": "IFCAXIS2PLACEMENT3D",
            "location": {
              "instanceId": 190333,
              "ifc": "IFCCARTESIANPOINT",
              "coordinates": [
                150.333353838582,
                0.184354494750211,
                0.666666666666667
              ]
            },
            "axis": {
              "instanceId": 19,
              "ifc": "IFCDIRECTION",
              "directionRatios": [
                0,
                0,
                1
              ]
            },
            "refDirection": {
              "instanceId": 13,
              "ifc": "IFCDIRECTION",
              "directionRatios": [
                -1,
                0,
                0
              ]
            }
          },
          "extrudedDirection": {
            "instanceId": 19,
            "ifc": "IFCDIRECTION",
            "directionRatios": [
              0,
              0,
              1
            ]
          },
          "depth": 35.4133333333333
        }
      ]
    }
  ]
}
```

Revision Control

URI: /ifcrestapi/roottrackings/revisioncontrols

```
#33=IFCTELECOMADDRESS(.OFFICE.,'office address','office address',
('111-111-1111'),('NA'),'NA',('keresh@gatech.edu'),
'www.kereshafsari.com',('None'));
#34=IFCACTORROLE(.ARCHITECT.,'RESEARCHER','PhD thesis');
#35= IFCPERSON('PCI','Afsari','Keresh',('NA'),('Ms'),
('NA'),(#34),(#33));
#37= IFCORGANIZATION('Gatech','GT','University',(#34),(#33));
#38= IFCPERSONANDORGANIZATION(#35,#37,(#34));
#5= IFCAPPLICATION(#37,'2017','Manual','Manual');
#41= IFCOWNERHISTORY(#38,#5,.READONLY.,.NOCHANGE.,
1407386573,#38,#5,1407386573);
```




```
{
  "instanceId": 41,
  "ifc": "IFCOWNERHISTORY",
  "owningUser": {
    "instanceId": 38,
    "ifc": "IFCPERSONANDORGANIZATION",
    "thePerson": {
      "instanceId": 35,
      "ifc": "IFCPERSON",
      "identification": "PCI",
      "familyName": "Afsari",
      "givenName": "Keresh",
      "middleNames": ["NA"],
      "prefixTitles": ["Ms"],
      "suffixTitles": ["NA"],
      "roles": [{
        "instanceId": 34,
        "ifc": "IFCACTORROLE",
        "role": "ARCHITECT",
        "userDefinedRole": "RESEARCHER",
        "description": "PhD thesis"
      }],
      "addresses": [{
        "instanceId": 33,
        "ifc": "IFCTELECOMADDRESS",
        "purpose": "OFFICE",
        "description": "office address",
        "userDefinedPurpose": "office address",
        "telephoneNumbers": ["111-111-1111"],
        "facsimileNumbers": ["NA"],
        "pagerNumber": "NA",
        "electronicMailAddresses": ["keresh@gatech.edu"],
        "wwwHomePageURL": "www.kereshafsari.com",
        "messagingIDs": ["None"]
      }]
    },
    "theOrganization": {
      "instanceId": 37,
      "ifc": "IFCORGANIZATION",
      "identification": "Gatech",
      "name": "GT",
      "description": "University",
      "roles": [{
        "instanceId": 34,
        "ifc": "IFCACTORROLE",
        "role": "ARCHITECT",
        "userDefinedRole": "RESEARCHER",
        "description": "PhD thesis"
      }],
      "addresses": [{
        "instanceId": 33,
        "ifc": "IFCTELECOMADDRESS",
        "purpose": "OFFICE",
        "description": "office address",
        "userDefinedPurpose": "office address",
        "telephoneNumbers": ["111-111-1111"],
        "facsimileNumbers": ["NA"],
        "pagerNumber": "NA",
        "electronicMailAddresses": ["keresh@gatech.edu"],
        "wwwHomePageURL": "www.kereshafsari.com",
        "messagingIDs": ["None"]
      }]
    },
    "roles": [{
      "instanceId": 34,
      "ifc": "IFCACTORROLE",
      "role": "ARCHITECT",
      "userDefinedRole": "RESEARCHER",
      "description": "PhD thesis"
    }]
  },
  "owningApplication": {
    "instanceId": 5,
    "ifc": "IFCAPPLICATION",
    "applicationDeveloper": {
      "instanceId": 37,
      "ifc": "IFCORGANIZATION",
      "identification": "Gatech",
      "name": "GT",
      "description": "University",
      "roles": [{
        "instanceId": 34,
        "ifc": "IFCACTORROLE",
        "role": "ARCHITECT",
```

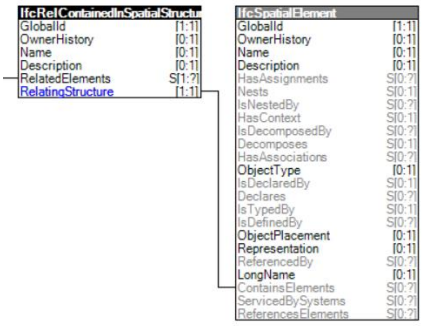
```
      "userDefinedRole": "RESEARCHER",
      "description": "PhD thesis"
    }],
    "addresses": [{
      "instanceId": 33,
      "ifc": "IFCTELECOMADDRESS",
      "purpose": "OFFICE",
      "description": "office address",
      "userDefinedPurpose": "office address",
      "telephoneNumbers": ["111-111-1111"],
      "facsimileNumbers": ["NA"],
      "pagerNumber": "NA",
      "electronicMailAddresses": ["keresh@gatech.edu"],
      "wwwHomePageURL": "www.kereshafsari.com",
      "messagingIDs": ["None"]
    }]
  },
  "version": "2017",
  "applicationFullName": "Manual",
  "applicationIdentifier": "Manual"
},
"state": "READONLY",
"changeAction": "NOCHANGE",
"lastModifiedDate": 1407386573,
"lastModifyingUser": {
  "instanceId": 38,
  "ifc": "IFCPERSONANDORGANIZATION",
  "thePerson": {
    "instanceId": 35,
    "ifc": "IFCPERSON",
    "identification": "PCI",
    "familyName": "Afsari",
    "givenName": "Keresh",
    "middleNames": ["NA"],
    "prefixTitles": ["Ms"],
    "suffixTitles": ["NA"],
    "roles": [{
      "instanceId": 34,
      "ifc": "IFCACTORROLE",
      "role": "ARCHITECT",
      "userDefinedRole": "RESEARCHER",
      "description": "PhD thesis"
    }],
    "addresses": [{
      "instanceId": 33,
      "ifc": "IFCTELECOMADDRESS",
      "purpose": "OFFICE",
      "description": "office address",
      "userDefinedPurpose": "office address",
      "telephoneNumbers": ["111-111-1111"],
      "facsimileNumbers": ["NA"],
      "pagerNumber": "NA",
      "electronicMailAddresses": ["keresh@gatech.edu"],
      "wwwHomePageURL": "www.kereshafsari.com",
      "messagingIDs": ["None"]
    }]
  },
  "theOrganization": {
    "instanceId": 37,
    "ifc": "IFCORGANIZATION",
    "identification": "Gatech",
    "name": "GT",
    "description": "University",
    "roles": [{
      "instanceId": 34,
      "ifc": "IFCACTORROLE",
      "role": "ARCHITECT",
      "userDefinedRole": "RESEARCHER",
      "description": "PhD thesis"
    }],
    "addresses": [{
      "instanceId": 33,
      "ifc": "IFCTELECOMADDRESS",
      "purpose": "OFFICE",
      "description": "office address",
      "userDefinedPurpose": "office address",
      "telephoneNumbers": ["111-111-1111"],
      "facsimileNumbers": ["NA"],
      "pagerNumber": "NA",
      "electronicMailAddresses": ["keresh@gatech.edu"],
      "wwwHomePageURL": "www.kereshafsari.com",
      "messagingIDs": ["None"]
    }]
  },
  "roles": [{
    "instanceId": 34,
    "ifc": "IFCACTORROLE",
    "role": "ARCHITECT",
    "userDefinedRole": "RESEARCHER",
    "description": "PhD thesis"
  }]
},
"lastModifyingApplication": {
```

```
  "instanceId": 5,
  "ifc": "IFCAPPLICATION",
  "applicationDeveloper": {
    "instanceId": 37,
    "ifc": "IFCORGANIZATION",
    "identification": "Gatech",
    "name": "GT",
    "description": "University",
    "roles": [{
      "instanceId": 34,
      "ifc": "IFCACTORROLE",
      "role": "ARCHITECT",
      "userDefinedRole": "RESEARCHER",
      "description": "PhD thesis"
    }],
    "addresses": [{
      "instanceId": 33,
      "ifc": "IFCTELECOMADDRESS",
      "purpose": "OFFICE",
      "description": "office address",
      "userDefinedPurpose": "office address",
      "telephoneNumbers": ["111-111-1111"],
      "facsimileNumbers": ["NA"],
      "pagerNumber": "NA",
      "electronicMailAddresses": ["keresh@gatech.edu"],
      "wwwHomePageURL": "www.kereshafsari.com",
      "messagingIDs": ["None"]
    }]
  },
  "version": "2017",
  "applicationFullName": "Manual",
  "applicationIdentifier": "Manual"
},
"creationDate": 1407386573
}
```


Spatial Containment

URI: /ifcrestapi/connectivities/spatialcontainments

```
#123= IFCBUILDINGSTOREY('1GZlOYvKP22QZKG5xdW5f7',#41,'Basement',
$,'Basement',#121,$,'Basement',$,0.666666666666667);
#4895= IFCRELCONTAINEDINSPATIALSTRUCTURE('2iiyvJC6z28xpx2jnhX4oI',
#41,$,$,(#201,#450,#565,#639,#1167,#3293,#3847,#4419,#4603999,
#46039996,#4603888,#46039995),#123);
```

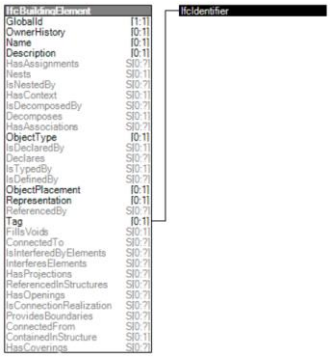



```
{
  "instanceId": 4895,
  "ifc": "IFCRELCONTAINEDINSPATIALSTRUCTURE",
  "globalId": "2iiyvJC6z28xpx2jnhX4oI",
  "ownerHistory": {
    "instanceId": 41
  },
  "name": null,
  "description": null,
  "relatedElements": [
    {"instanceId": 201},
    {"instanceId": 450},
    {"instanceId": 565},
    {"instanceId": 639},
    {"instanceId": 1167},
    {"instanceId": 3293},
    {"instanceId": 3847},
    {"instanceId": 4419},
    {"instanceId": 4603999},
    {"instanceId": 46039996},
    {"instanceId": 4603888},
    {"instanceId": 46039995}
  ],
  "relatingStructure":{
    "instanceId": 123
  }
}
```


PCI-067 Piece Mark of Building Element

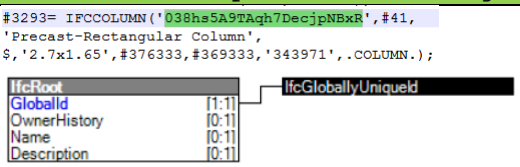
URI: /ifcrestapi/pci067

```
#3293= IFCCOLUMN('038hs5A9TAqh7DecjpNBxR',#41,
'Precast-Rectangular Column',
$,'2.7x1.65',#376333,#369333,'343971',.COLUMN.);
```

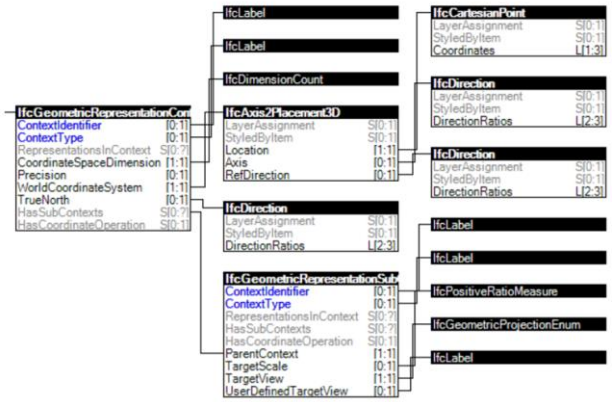



```
{
  "instanceId": 3293,
  "tag": "343971"
}
```

Identity	
URI: /ifcrestapi/roottrackings/identities	

	{
	"instanceId": 3293,
	"globalId": "038hs5A9TAqh7DecjpNBxR"
	}

Project Context	
URI: /ifcrestapi/projects/projectcontexts	

	{
	"instanceId": 96,
	"ifc": "IFCGEOMETRICREPRESENTATIONSUBCONTEXT",
	"contextIdentifier": "Body",
	"contextType": "Model",
	"parentContext": {
	"instanceId": 91,
	"ifc": "IFCGEOMETRICREPRESENTATIONCONTEXT",
	"contextIdentifier": "Body",
	"contextType": "Model",
	"coordinateSpaceDimension": 3,
	"precision": 0.0001,
	"worldCoordinateSystem": {
	"instanceId": 88,
	"ifc": "IFCAXIS2PLACEMENT3D",
	"location": {
	"instanceId": 6,
	"ifc": "IFCCARTESIANPOINT",
	"coordinates": [
	0,
	0,
	0
	]
	},
	"axis": null,
	"refDirection": null
	},
	"trueNorth": {
	"instanceId": 89,
	"ifc": "IFCDIRECTION",
	"directionRatios": [
	0,
	1
	]
	}
	},
	"targetScale": 1,
	"targetView": "MODEL_VIEW",
	"userDefinedTargetView": "perspective"
	}

The relationships and links among the resources for the precast column in this exchange are diagrammed in Figure 76. In Table 39, some REST resources are shared among all other building elements or entities that refer to the same resource. For example, “Revision Control” and “Project Context” resources are used in all other objects that are referring to owner history data and geometric representation context data respectively. Also “Spatial Containment” resource shows the same data for all relating elements that are listed under “relatedElements” property.

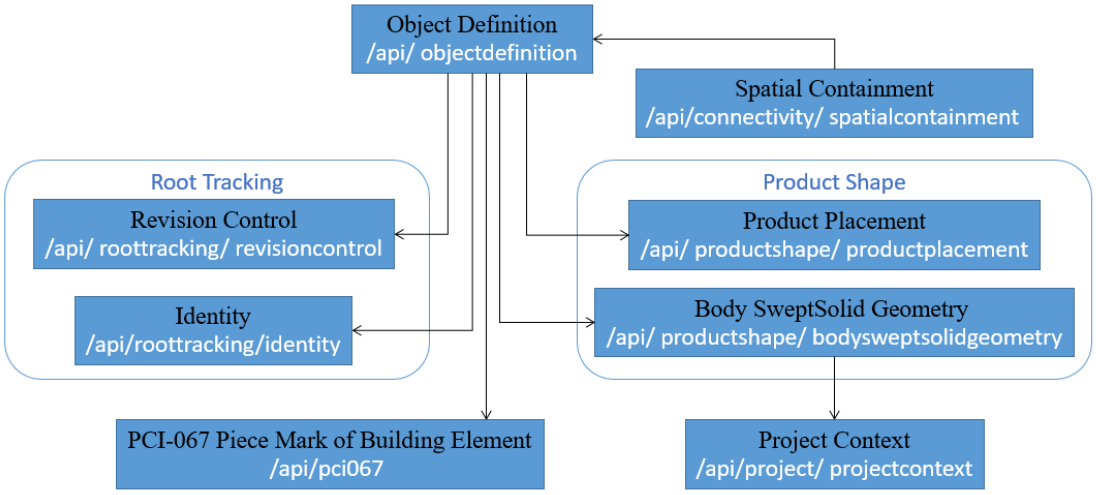


Figure 76 Links between the REST resources representing data for the precast column

The data for this precast concrete column instance corresponds to 8 sets of resources in 8 collections. Seven of these collections have a direct relationship to the IfcColumn entity itself and one of the collections which is the "Project Context" indirectly links to the column. In fact, "Project Context" concept definition that includes main geometric representation context of the project in IFC4 specification is vague. The concept definition focuses on the project and deals with IfcGeometricRepresentationContext but at the same time, "Product Geometric Representation" concept also implements IfcGeometricRepresentationContext entity. Therefore, the "Project Context" concept and REST collection indirectly links with the building elements such as column.

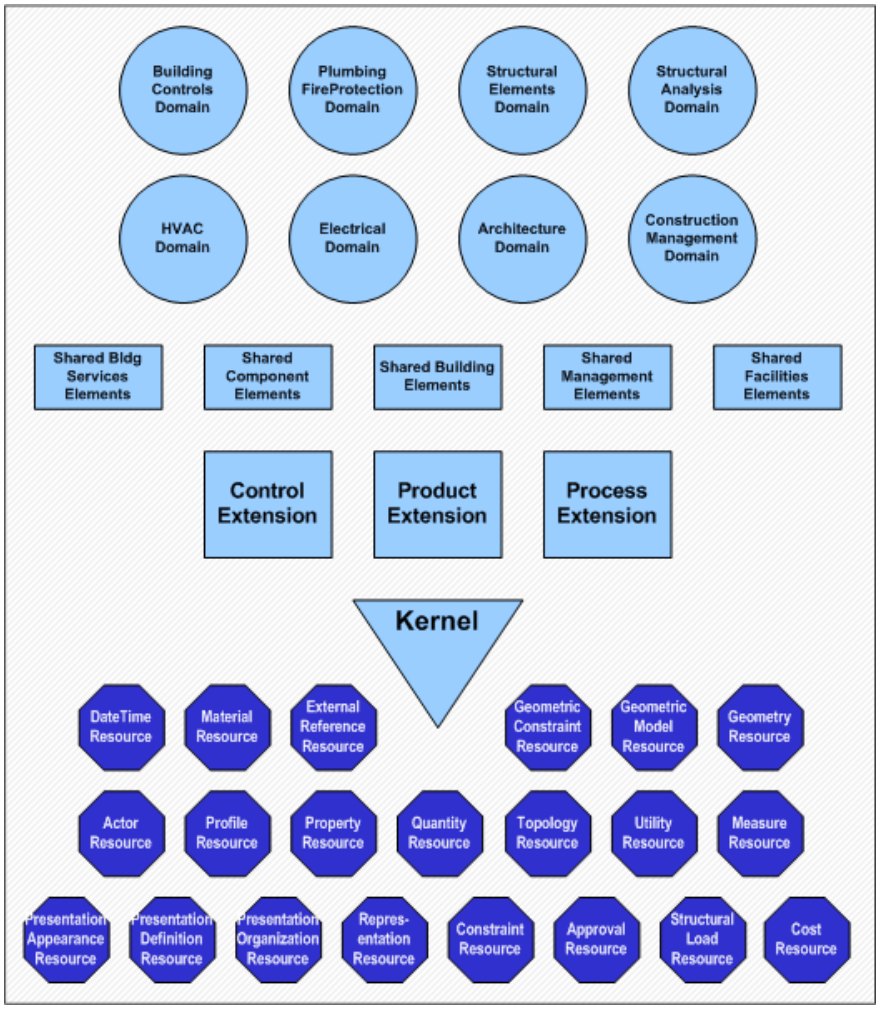


Figure 77 IFC architecture and its resource schema layer

As shown in Table 39 in the right column, the links in this implementation are managed through “instanceId” property of ifcJSON. But as explained in chapter 5, the “href” property can perform as direct resource links. GUID attribute in IFC specification cannot provide resource linking for REST collections because some collections i.e. IFC concepts may not have IFC entities with GUID attribute in them. In IFC specification, the entities that are defined in an IFC resource schema do not have GUIDs. The resource

schema layer of IFC architecture, shown in Figure 77 and highlighted in dark blue, does not carry a concept of identity. Unlike entities in other layers of IFC architecture, the entities belong to resource schema layer cannot exist independently, but can only exist if referenced by other entities that are sub-type of IfcRoot [60]. So, not every IFC entity includes GUID attribute and therefore, IFC concepts may not include entities with GUID.

For example, in IFC specification the concept definition for “Project Context” concept starts with IfcGeometricRepresentationContext entity, for “Revision Control” concept starts with IfcOwnerHistory entity, and for “Body Geometry” concept starts with IfcProductDefinitionShape entity. None of these entities include a GUID attribute. Therefore, as shown in Table 39, REST resources in these two collections do not contain any IFC-specific identity property thus, should use either instanceId or href property to provide the resource links.

Another example of REST resource mapping is shown in Table 40 that specifies the mapping guide for the precast concrete double tee. Each slab in this BIM model contains three double tees. 9 REST resource collections contain the data for the double tee: “Object Definition”, “Product Placement”, “Body SweptSolid Geometry”, “PCI-069”, “Revision Control”, “Element Composition”, “PCI-067”, “Identity”, and “Project Context”. These collections that contain data for a double tee slab and their relationship are illustrated in Figure 78. Two collections i.e. “PCI-069” and “Project Context” indirectly link to the slab data in “Object Definition” collection while other collections have explicit and direct relationship to the “Object Dentition” collection.

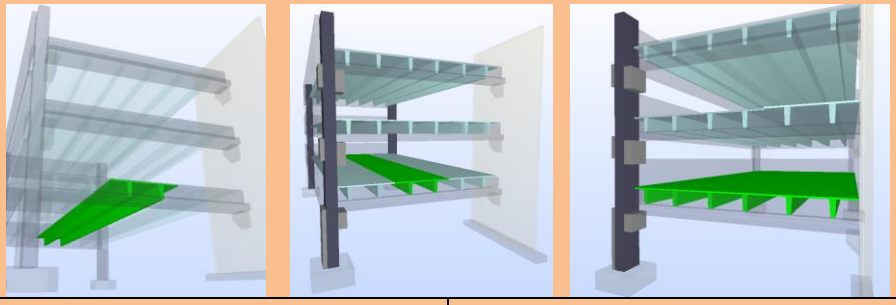
Table 40 Mapping guide for precast concrete double tee

Schema IFC4 Add2

MVD: Precast Concrete

Exchange Model: EMPC.1

Data: Precast Concrete Slab with Three Double Tees

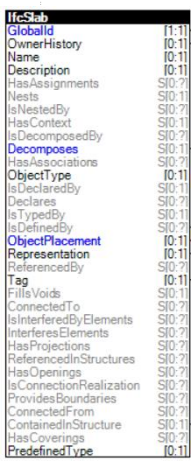


IFC SPF Data and Diagram	ifcJSON REST Resources

Object Definition (ifcbuildingelement)

URI: `/ifcrestapi/objectdefinitions/ifcproducts/ifcelements/Ifcbuildingelements`

```
#1932= IFCSLAB('2NwynIVsfAOhBpTS6WCoWk',#41,
'Precast-Concrete Double Tees',$,
'Precast-Concrete Double Tees',#1887,#1930,
'276410',.BASESLAB.);
```



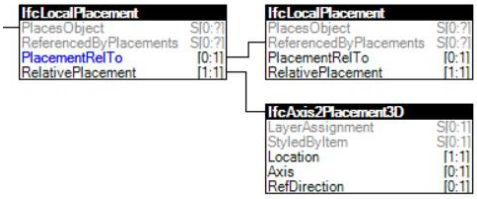

```
{
  "instanceId": 1932,
  "ifc": "IFCSLAB",
  "globalId": "2NwynIVsfAOhBpTS6WCoWk",
  "ownerHistory": {
    "instanceId": 41
  },
  "name": "Precast-Concrete Double Tees",
  "description": null,
  "objectType": "Precast-Concrete Double Tees",
  "objectPlacement": {
    "instanceId": 1887
  },
  "representation": {
    "instanceId": 1930
  },
  "tag": "276410",
  "predefinedType": "BASESLAB"
}
```


Product Placement

URI: `/ifcrestapi/productshapes/productplacements`

```
#17= IFCDIRECTION((0.,-1.,0.));
#19= IFCDIRECTION((0.,0.,1.));
#6= IFCCARTESIANPOINT((0.,0.,0.));
#4813= IFCAXIS2PLACEMENT3D(#6,$,$);
#4814= IFCLOCALPLACEMENT($,#4813);
#31= IFCAXIS2PLACEMENT3D(#6,$,$);
#32= IFCLOCALPLACEMENT(#4814,#31);
#137= IFCCARTESIANPOINT((0.,0.,32.));
#139= IFCAXIS2PLACEMENT3D(#137,$,$);
#140= IFCLOCALPLACEMENT(#32,#139);
#1884= IFCCARTESIANPOINT
((174.999333333332,59.1874897473753,0.));
#1886= IFCAXIS2PLACEMENT3D(#1884,#19,#17);
#1887= IFCLOCALPLACEMENT(#140,#1886);
```


```
{
  "instanceId": 1887,
  "ifc": "IFCLOCALPLACEMENT",
  "placementRelTo": {
    "instanceId": 140
  },
  "relativePlacement": {
    "instanceId": 1886,
    "ifc": "IFCAXIS2PLACEMENT3D",
    "location": {
      "instanceId": 1884,
      "ifc": "IFCCARTESIANPOINT",
      "coordinates": [174.999333333332, 59.1874897473753, 0]
    },
    "axis": {
      "instanceId": 19,
      "ifc": "IFCDIRECTION",
      "directionRatios": [0, 0, 1]
    },
```



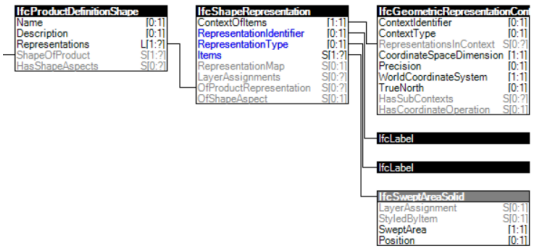

```
      "refDirection": {
        "instanceId": 17,
        "ifc": "IFCDIRECTION",
        "directionRatios": [0, -1, 0]
      }
    }
  }, {
    "instanceId": 140,
    "ifc": "IFCLOCALPLACEMENT",
    "placementRelTo": {
      "instanceId": 32
    },
    "relativePlacement": {
      "instanceId": 139,
      "ifc": "IFCAXIS2PLACEMENT3D",
      "location": {
        "instanceId": 137,
        "ifc": "IFCCARTESIANPOINT",
        "coordinates": [0, 0, 32]
      },
      "axis": null,
      "refDirection": null
    }
  },{
    "instanceId": 32,
    "ifc": "IFCLOCALPLACEMENT",
    "placementRelTo": {
      "instanceId": 4814
    },
    "relativePlacement": {
      "instanceId": 31,
      "ifc": "IFCAXIS2PLACEMENT3D",
      "location": {
        "instanceId": 6,
        "ifc": "IFCCARTESIANPOINT",
        "coordinates": [0, 0, 0]
      },
      "axis": null,
      "refDirection": null
    }
  },{
    "instanceId": 4814,
    "ifc": "IFCLOCALPLACEMENT",
    "placementRelTo": null,
    "relativePlacement": {
      "instanceId": 4813,
      "ifc": "IFCAXIS2PLACEMENT3D",
      "location": {
        "instanceId": 6,
        "ifc": "IFCCARTESIANPOINT",
        "coordinates": [0, 0, 0]
      },
      "axis": null,
      "refDirection": null
    }
  }
```


Body SweptSolid Geometry

URL: `/ifcrestapi/productshapes/bodysweptsolidgeometries`

```
#11= IFCDIRECTION((1.,0.,0.));
#19= IFCDIRECTION((0.,0.,1.));
#21= IFCDIRECTION((0.,0.,-1.));
#1888= IFCCARTESIANPOINT((-0.833333333333314,-4.92499999999995));
#1890= IFCCARTESIANPOINT((-0.633333333333315,-4.92499999999995));
#1892= IFCCARTESIANPOINT((-0.633333333333322,-2.72499999999994));
#1894= IFCCARTESIANPOINT((1.36666666666668,-2.62499999999994));
#1896= IFCCARTESIANPOINT((1.36666666666668,-2.12499999999994));
#1898= IFCCARTESIANPOINT((-0.633333333333326,-2.02499999999995));
#1900= IFCCARTESIANPOINT((-0.633333333333336,1.97500000000005));
#1902= IFCCARTESIANPOINT((1.36666666666666,2.07500000000005));
#1904= IFCCARTESIANPOINT((1.36666666666666,2.57500000000005));
#1906= IFCCARTESIANPOINT((-0.633333333333347,2.67500000000004));
#1908= IFCCARTESIANPOINT((-0.633333333333347,5.02500000000003));
#1910= IFCCARTESIANPOINT((-0.833333333333346,5.02500000000003));
#1912= IFCPOLYLINE((#1888,#1890,#1892,#1894,#1896,#1898,#1900,
#1902,#1904,#1906,#1908,#1910,#1888));
#1914= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'48"x43"',#1912);
#1915= IFCCARTESIANPOINT((0.,0.0249999999999857,-0.83333333333335));
#1917= IFCAXIS2PLACEMENT3D(#1915,#11,#21);
#1918= IFCEXTRUDEDAREASOLID(#1914,#1917,#19,59.8374897473758);
#1922= IFCSHAPEREPRESENTATION(#96,'Body','SweptSolid',(#1918));
#1930= IFCPRODUCTDEFINITIONSHAPE($,$,(#1922));
```


```
{
  "instanceId": 1930,
  "ifc": "IFCPRODUCTDEFINITIONSHAPE",
  "name": null,
  "description": null,
  "representations": [{
    "instanceId": 1922,
    "ifc": "IFCSHAPEREPRESENTATION",
    "contextOfItems": {
      "instanceId": 96
    },
    "representationIdentifier": "Body",
    "representationType": "SweptSolid",
    "items": [{
      "instanceId": 1918,
      "ifc": "IFCEXTRUDEDAREASOLID",
      "sweptArea": {
        "instanceId": 1914,
      },
      "position": {
        "instanceId": 1917,
        "ifc": "IFCAXIS2PLACEMENT3D",
        "location": {
          "instanceId": 1915,
          "ifc": "IFCCARTESIANPOINT",
          "coordinates": [0, 0.0249999999999857, -0.83333333333335]
        },
        "axis": {
          "instanceId": 11,
```

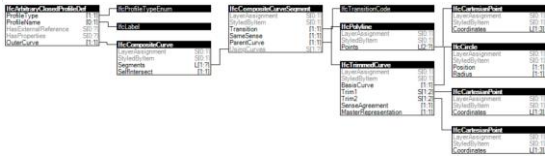



```
        "ifc": "IFCDIRECTION",
        "directionRatios": [1, 0, 0]
      },
      "refDirection": {
        "instanceId": 21,
        "ifc": "IFCDIRECTION",
        "directionRatios": [0, 0, -1]
      }
    },
    "extrudedDirection": {
      "instanceId": 19,
      "ifc": "IFCDIRECTION",
      "directionRatios": [0, 0, 1]
    },
    "depth": 59.8374897473758
  }]
 }]
}
```


PCI-069 Arbitrary Profile

URI: `/ifcrestapi/pci069`

```
#1888= IFCCARTESIANPOINT((-0.833333333333314,-4.92499999999995));
#1890= IFCCARTESIANPOINT((-0.633333333333315,-4.92499999999995));
#1892= IFCCARTESIANPOINT((-0.633333333333322,-2.72499999999994));
#1894= IFCCARTESIANPOINT((1.36666666666668,-2.62499999999994));
#1896= IFCCARTESIANPOINT((1.36666666666668,-2.12499999999994));
#1898= IFCCARTESIANPOINT((-0.633333333333326,-2.02499999999995));
#1900= IFCCARTESIANPOINT((-0.633333333333336,1.97500000000005));
#1902= IFCCARTESIANPOINT((1.36666666666666,2.07500000000005));
#1904= IFCCARTESIANPOINT((1.36666666666666,2.57500000000005));
#1906= IFCCARTESIANPOINT((-0.633333333333347,2.67500000000004));
#1908= IFCCARTESIANPOINT((-0.633333333333347,5.02500000000003));
#1910= IFCCARTESIANPOINT((-0.833333333333346,5.02500000000003));
#1912= IFCPOLYLINE((#1888,#1890,#1892,#1894,#1896,#1898,#1900,
#1902,#1904,#1906,#1908,#1910,#1888));
#1914= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'48"x43"',#1912);
```




```
{
  "instanceId": 1914,
  "ifc": "IFCARBITRARYCLOSEDPROFILEDEF",
  "profileType": "AREA",
  "profileName": "48x43",
  "outerCurve": {
    "instanceId": 1912,
    "ifc": "IFCPOLYLINE",
    "points": [{
      "instanceId": 1888,
      "ifc": "IFCCARTESIANPOINT",
      "coordinates": [-0.833333333333314, -4.92499999999995]
    }, {
      "instanceId": 1890,
      "coordinates": [-0.633333333333315, -4.92499999999995]
    }, {
      "instanceId": 1892,
      "coordinates": [-0.633333333333322, -2.72499999999994]
    }, {
      "instanceId": 1894,
      "coordinates": [1.36666666666668, -2.62499999999994]
    }, {
      "instanceId": 1896,
      "coordinates": [1.36666666666668, -2.12499999999994]
    }, {
      "instanceId": 1898,
      "coordinates": [-0.633333333333326, -2.02499999999995]
    }, {
      "instanceId": 1900,
      "coordinates": [-0.633333333333336, 1.97500000000005]
    }, {
      "instanceId": 1902,
      "coordinates": [1.36666666666666, 2.07500000000005]
    }, {
      "instanceId": 1904,
      "coordinates": [1.36666666666666, 2.57500000000005]


    }, {
      "instanceId": 1906,
      "coordinates": [-0.633333333333347, 2.67500000000004]
    }, {
      "instanceId": 1908,
      "coordinates": [-0.633333333333347, 5.02500000000003]
    }, {
      "instanceId": 1910,
      "coordinates": [0.833333333333346, 5.02500000000003]
    }, {
      "instanceId": 1888,
      "coordinates": [-0.833333333333314, -4.92499999999995]
    }]
  }
}
```


Revision Control

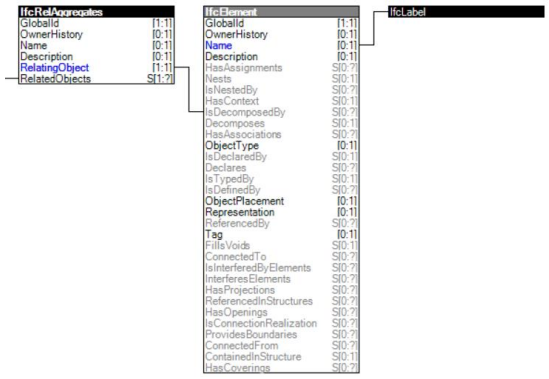
URI: `/ifcrestapi/roottrackings/revisioncontrols`

See Table 39- Revision Control	See Table 39- Revision Control

Element Composition

URI: `/ifcrestapi/compositions/elementcompositions`

```
#47970= IFCRELAGGREGATES('1UZGCFuIzAPRf3$5t793mR',#41,
$,$,#45330,(#1656,#1814,#1932));
```

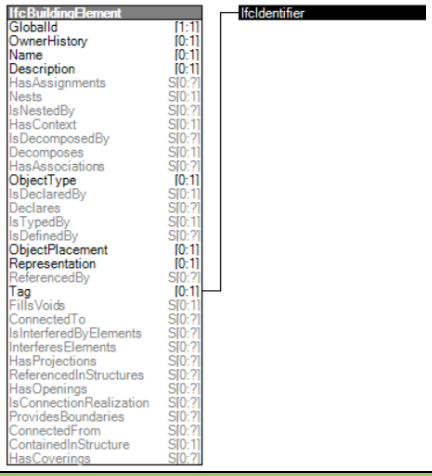



```
{
  "instanceId": 47970,
  "ifc": "IFCRELAGGREGATES",
  "globalId": "1UZGCFuIzAPRf3$5t793mR",
  "ownerHistory": {
    "instanceId": 41
  },
  "name": null,
  "description": null,
  "relatingObject": {
    "instanceId": 45330
  },
  "relatedObjects": [
    {
      "instanceId": 1656
    },
    {
      "instanceId": 1814
    },
    {
      "instanceId": 1932
    }
  ]
}
```


PCI-067 Piece Mark of Building Element

URI: `/ifcrestapi/pci067`

```
#1932= IFCSLAB('2NwynIVsfAOhBpTS6WCoWk',#41,
'Precast-Concrete Double Tees',$,
'Precast-Concrete Double Tees',
#1887,#1930,'276410',.BASESLAB.);
```

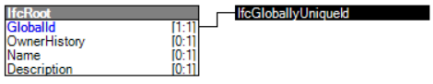



```
{
  "instanceId": 1932,
  "tag": "276410"
}
```


Identity

URI: `/ifcrestapi/roottrackings/identities`

```
#1932= IFCSLAB('2NwynIVsfAOhBpTS6WCoWk',#41,
'Precast-Concrete Double Tees',$,
'Precast-Concrete Double Tees',
#1887,#1930,'276410',.BASESLAB.);
```




```
{
  "instanceId": 1932,
  "globalId": "2NwynIVsfAOhBpTS6WCoWk"
}
```


Project Context

URI: `/ifcrestapi/projects/projectcontexts`

See Table 39- Project Context	See Table 39- Project Context

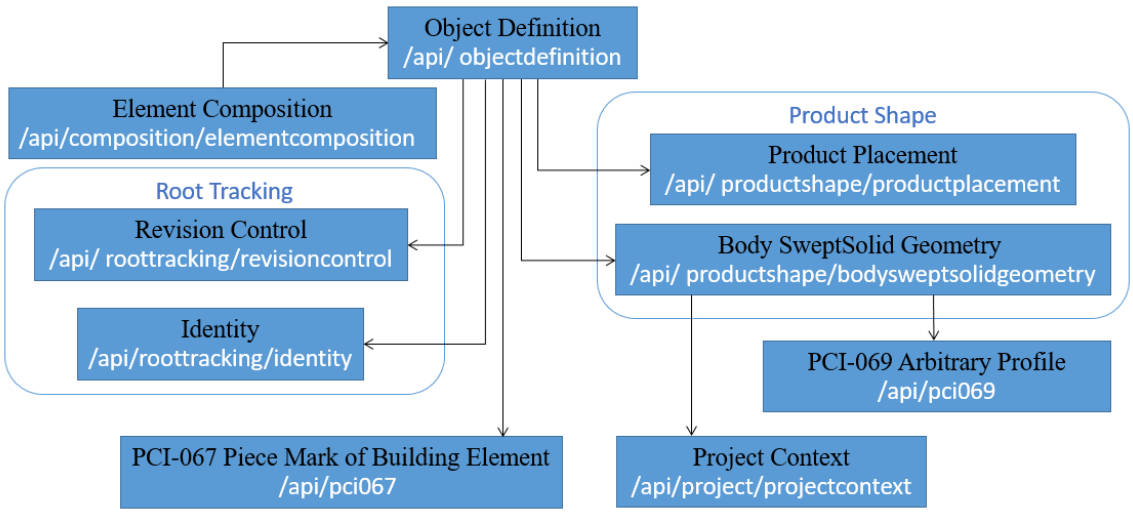


Figure 78 Links between the REST resources representing data for double tee slab

The "Object Definition" REST collection here is narrowed down to two of its sub-collections as "IfcElement" and "IfcSpatialElement". Accordingly, the "IfcElement" collection is narrowed down to "IfcBuildingElement" sub-collection and "IfcElementAssembly" sub-collection. The relationship between main and sub-collection was previously shown in Figure 39.

Table 41 REST resources in IfcBuildingElement sub-collection

REST Sub-Collection: IfcBuildingElement		
REST Main Collection: Object Definition		
URI: `/ifcrestapi/objectdefinitions/ifcproducts/ifcelements/ifcbuildingelements`		
IFC Entity	Count	REST Resources
IfcColumn	3	{ "instanceId": 3293, "ifc": "IFCCOLUMN", "globalId": "038hs5A9TAqh7DecjpNBxR", "ownerHistory": {"instanceId": 41}, "name": "Precast-Rectangular Column", "description": null, "objectType": "2.7x1.65", "objectPlacement": {"instanceId": 376333}, "representation": {"instanceId": 369333}, "tag": "343971", "predefinedType": "COLUMN" },

		{
		"instanceId": 3847,
		"ifc": "IFCCOLUMN",
		"globalId": "038hs5A9TAqh7DecjpNC4u",
		"ownerHistory": {"instanceId": 41},
		"name": "Precast-Rectangular Column",
		"description": null,
		"objectType": "2.7x1.65",
		"objectPlacement": {"instanceId": 3846},
		"representation": {"instanceId": 3841},
		"tag": "344128",
		"predefinedType": "COLUMN"
		},
		{
		"instanceId": 4419,
		"ifc": "IFCCOLUMN",
		"globalId": "2Qxzh7CYb8rPETGRVxNE9m",
		"ownerHistory": {"instanceId": 41},
		"name": "Precast-Rectangular Column",
		"description": null,
		"objectType": "2.7x1.65",
		"objectPlacement": {"instanceId": 409222},
		"representation": {"instanceId": 402222},
		"tag": "373122",
		"predefinedType": "COLUMN"
		}

IfcFooting	4	{
		"instanceId": 450,
		"ifc": "IFCFOOTING",
		"globalId": "1avE5ZL014oxIzubMPI4k$",
		"ownerHistory": {"instanceId": 41},
		"name": "Footing-Rectangular",
		"description": null,
		"objectType": "58 x 58 x 25",
		"objectPlacement": {"instanceId": 428},
		"representation": {"instanceId": 448},
		"tag": "188793",
		"predefinedType": "PAD_FOOTING"
		},
		{
		"instanceId": 565,
		"ifc": "IFCFOOTING",
		"globalId": "1avE5ZL014oxIzubMPI4jy",
		"ownerHistory": {"instanceId": 41},
		"name": "Footing-Rectangular",
		"description": null,
		"objectType": "58 x 58 x 25",
		"objectPlacement": {"instanceId": 549},
		"representation": {"instanceId": 563},
		"tag": "188858",
		"predefinedType": "PAD_FOOTING"
		},
		{
		"instanceId": 639,
		"ifc": "IFCFOOTING",
		"globalId": "1avE5ZL014oxIzubMPI4v0",
		"ownerHistory": {"instanceId": 41},
		"name": "Footing-Rectangular",
		"description": null,
		"objectType": "58 x 58 x 25",
		"objectPlacement": {"instanceId": 623},
		"representation": {"instanceId": 637},
		"tag": "189574",
		"predefinedType": "PAD_FOOTING"
		},
		{
		"instanceId": 1167,
		"ifc": "IFCFOOTING",
		"globalId": "3O1XrN3ar2X8cWuEdJAGlu",
		"ownerHistory": {"instanceId": 41},
		"name": "Wall Foundation",
		"description": null,
		"objectType": "Wall Foundation",
		"objectPlacement": {"instanceId": 1145},
		"representation": {"instanceId": 1165},
		"tag": "240945",
		"predefinedType": "STRIP_FOOTING"
		}

IfcWall	1	{
		"instanceId": 201,
		"ifc": "IFCWALL",
		"globalId": "3jxK8DUYD9pRyTsJ2VV8j2",
		"ownerHistory": {"instanceId": 41},
		"name": "Basic Wall",
		"description": null,
		"objectType": "Precast Wall",
		"objectPlacement": {"instanceId": 171},
		"representation": {"instanceId": 197},
		"tag": "185436",
		"predefinedType": "STANDARD"
		}

| IfcBeam | 9 |
```
{
  "instanceId": 726,
  "ifc": "IFCBEAM",
  "globalId": "35aaXitMrAUBLn3UfJ2EDm",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Rectangular Beam",
  "description": null,
  "objectType": "Precast-Rectangular Beam"
  "objectPlacement": {"instanceId": 697},
  "representation": {"instanceId": 724},
  "tag": "217722",
  "predefinedType": "BEAM"
},
{
  "instanceId": 873,
  "ifc": "IFCBEAM",
  "globalId": "3Ff7uirwb5ovirJqJqkn2t",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Rectangular Beam",
  "description": null,
  "objectType": "Precast-Rectangular Beam"
  "objectPlacement": {"instanceId": 851},
  "representation": {"instanceId": 871},
  "tag": "229829",
  "predefinedType": "BEAM"
},
{
  "instanceId": 972,
  "ifc": "IFCBEAM",
  "globalId": "3O1XrN3ar2X8cWuEdJAGno",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Rectangular Beam",
  "description": null,
  "objectType": "Precast-Rectangular Beam",
  "objectPlacement": {"instanceId": 949},
  "representation": {"instanceId": 970},
  "tag": "240315",
  "predefinedType": "BEAM"
},
{
  "instanceId": 1071,
  "ifc": "IFCBEAM",
  "globalId": "3O1XrN3ar2X8cWuEdJAGsx",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Rectangular Beam",
  "description": null,
  "objectType": "Precast-Rectangular Beam",
  "objectPlacement": {"instanceId": 1048},
  "representation": {"instanceId": 1069},
  "tag": "240498",
  "predefinedType": "BEAM"
},
{
  "instanceId": 1286,
  "ifc": "IFCBEAM",
  "globalId": "2iucYarOD7ThaZL2GNzPG3",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Inverted Tee",
  "description": null,
  "objectType": "Precast-Inverted Tee",
  "objectPlacement": {"instanceId": 1249},
  "representation": {"instanceId": 1284},
  "tag": "243091",
  "predefinedType": "BEAM"
},
{
  "instanceId": 1420,
  "ifc": "IFCBEAM",
  "globalId": "2iucYarOD7ThaZL2GNzPG7",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Inverted Tee",
  "description": null,
  "objectType": "Precast-Inverted Tee",
  "objectPlacement": {"instanceId": 1383},
  "representation": {"instanceId": 1418},
  "tag": "243095",
  "predefinedType": "BEAM"
},
{
  "instanceId": 1531,
  "ifc": "IFCBEAM",
  "globalId": "2iucYarOD7ThaZL2GNzPTp",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Inverted Tee",
  "description": null,
  "objectType": "Precast-Inverted Tee",
  "objectPlacement": {"instanceId": 1494},
  "representation": {"instanceId": 1529},
  "tag": "243427",
  "predefinedType": "BEAM"
},
```
 |
|---|---|---|

		{
		"instanceId": 2027,
		"ifc": "IFCBEAM",
		"globalId": "17_aE2Mjv8phyra7zxiXKJ",
		"ownerHistory": {"instanceId": 41},
		"name": "Precast-Rectangular Beam",
		"description": null,
		"objectType": "Precast-Rectangular Beam",
		"objectPlacement": {"instanceId": 2005},
		"representation": {"instanceId": 2025},
		"tag": "281622",
		"predefinedType": "BEAM"
		},
		{
		"instanceId": 2124,
		"ifc": "IFCBEAM",
		"globalId": "17_aE2Mjv8phyra7zxiXLB",
		"ownerHistory": {"instanceId": 41},
		"name": "Precast-Rectangular Beam",
		"description": null,
		"objectType": "Precast-Rectangular Beam",
		"objectPlacement": {"instanceId": 2102},
		"representation": {"instanceId": 2122},
		"tag": "281678",
		"predefinedType": "BEAM"
		}

IfcSlab	9	{ "instanceId": 1656, "ifc": "IFCSLAB", "globalId": "2NwynIVsfAOhBpTS6WCoWY", "ownerHistory": {"instanceId": 41}, "name": "Precast-Concrete Double Tees", "description": null, "objectType": "Precast-Concrete Double Tees", "objectPlacement": {"instanceId": 1605}, "representation": {"instanceId": 1654}, "tag": "276406", "predefinedType": "BASESLAB" }, { "instanceId": 1814, "ifc": "IFCSLAB", "globalId": "2NwynIVsfAOhBpTS6WCoWi", "ownerHistory": {"instanceId": 41}, "name": "Precast-Concrete Double Tees", "description": null, "objectType": "Precast-Concrete Double Tees", "objectPlacement": {"instanceId": 1769}, "representation": {"instanceId": 1812}, "tag": "276408", "predefinedType": "BASESLAB" }, { "instanceId": 1932, "ifc": "IFCSLAB", "globalId": "2NwynIVsfAOhBpTS6WCoWk", "ownerHistory": {"instanceId": 41}, "name": "Precast-Concrete Double Tees", "description": null, "objectType": "Precast-Concrete Double Tees", "objectPlacement": {"instanceId": 1887}, "representation": {"instanceId": 1930}, "tag": "276410", "predefinedType": "BASESLAB" }, { "instanceId": 2244, "ifc": "IFCSLAB", "globalId": "1yW8gtAEz7yRy519Q_7wpG", "ownerHistory": {"instanceId": 41}, "name": "Precast-Concrete Double Tees", "description": null, "objectType": "Precast-Concrete Double Tees", "objectPlacement": {"instanceId": 2199}, "representation": {"instanceId": 2242}, "tag": "285317", "predefinedType": "BASESLAB" }, { "instanceId": 2363, "ifc": "IFCSLAB", "globalId": "1yW8gtAEz7yRy519Q_7wpI", "ownerHistory": {"instanceId": 41}, "name": "Precast-Concrete Double Tees", "description": null, "objectType": "Precast-Concrete Double Tees", "objectPlacement": {"instanceId": 2318}, "representation": {"instanceId": 2361}, "tag": "285319", "predefinedType": "BASESLAB" },

```
{
  "instanceId": 22700,
  "ifc": "IFCSLAB",
  "globalId": "1yW8gtAEz7yRy519Q_7wpS",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Concrete Double Tees",
  "description": null,
  "objectType": "Precast-Concrete Double Tees",
  "objectPlacement": {"instanceId": 17100},
  "representation": {"instanceId": 22300},
  "tag": "285321",
  "predefinedType": "BASESLAB"
},
{
  "instanceId": 2640,
  "ifc": "IFCSLAB",
  "globalId": "1yW8gtAEz7yRy519Q_7wt5",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Concrete Double Tees",
  "description": null,
  "objectType": "Precast-Concrete Double Tees",
  "objectPlacement": {"instanceId": 2595},
  "representation": {"instanceId": 2638},
  "tag": "285584",
  "predefinedType": "BASESLAB"
},
{
  "instanceId": 2759,
  "ifc": "IFCSLAB",
  "globalId": "1yW8gtAEz7yRy519Q_7wt7",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Concrete Double Tees",
  "description": null,
  "objectType": "Precast-Concrete Double Tees",
  "objectPlacement": {"instanceId": 2714},
  "representation": {"instanceId": 2757},
  "tag": "285586",
  "predefinedType": "BASESLAB"
},
{
  "instanceId": 2879,
  "ifc": "IFCSLAB",
  "globalId": "1yW8gtAEz7yRy519Q_7wt1",
  "ownerHistory": {"instanceId": 41},
  "name": "Precast-Concrete Double Tees",
  "description": null,
  "objectType": "Precast-Concrete Double Tees",
  "objectPlacement": {"instanceId": 2834},
  "representation": {"instanceId": 2877},
  "tag": "285588",
  "predefinedType": "BASESLAB"
}
```


The “IfcBuildingElement” sub-collection contains all data related to IfcBuildingElement entities such as column, beam, slab, etc. Table 41 lists REST resources in “IfcBuildingElement” sub-collection. Also, the “IfcElementAssembly” sub-collection contains the data for any IfcElementAssembly entity in the BIM model. In fact, Table 41 captures all data related to “IfcBuildingElement” sub-collection based on “Object Definition” concept in IFC schema as well as links to corresponding collections. For example, “ownerHistory” property points to “Revision Control” collection,

“objectPlacement” property points to “Product Placement” collection, and “representation” property points to “Body SweptSolid Geometry” collection.

REST resources in “Spatial Containment” sub-collection is shown in Table 42. In this sub-collection, all data for the relationship of elements i.e. building elements and assemblies with a spatial structure element is specified.

Table 42 REST resources in Spatial Containment sub-collection

REST Sub-Collection: Spatial Containment		
REST Main Collection: Connectivity		
URI: `/ifcrestapi/connectivities/spatialcontainments`		
IFC Entity	Count	REST Resources

IfcRelContainedInSpatialStructure	4	{
		"instanceId": 4895,
		"ifc": "IFCRELCONTAINEDINSPATIALSTRUCTURE",
		"globalId": "2iiyvJC6z28xpx2jnhX4oI",
		"ownerHistory": {
		"instanceId": 41
		},
		"name": null,
		"description": null,
		"relatedElements": [
		{"instanceId": 201},
		{"instanceId": 450},
		{"instanceId": 565},
		{"instanceId": 639},
		{"instanceId": 1167},
		{"instanceId": 3293},
		{"instanceId": 3847},
		{"instanceId": 4419},
		{"instanceId": 4603999},
		{"instanceId": 46039996},
		{"instanceId": 4603888},
		{"instanceId": 46039995}
		],
		"relatingStructure":{
		"instanceId": 123
		}
		},
		{
		"instanceId": 4939,
		"ifc": "IFCRELCONTAINEDINSPATIALSTRUCTURE",
		"globalId": "2iiyvJC6z28xpx2jnhX4ps",
		"ownerHistory": {
		"instanceId": 41
		},
		"name": null,
		"description": null,
		"relatedElements": [
		{"instanceId": 1071},
		{"instanceId": 1286},
		{"instanceId": 2124},
		{"instanceId": 4503},
		{"instanceId": 4603},
		{"instanceId": 46039994}
		],
		"relatingStructure":{
		"instanceId": 129
		}
		},

```
{
  "instanceId": 4973,
  "ifc": "IFCRELCONTAINEDINSPATIALSTRUCTURE",
  "globalId": "2iiyvJC6z28xpx2jnhX4pF",
  "ownerHistory": {
    "instanceId": 41
  },
  "name": null,
  "description": null,
  "relatedElements": [
    {"instanceId": 972},
    {"instanceId": 1420},
    {"instanceId": 2027},
    {"instanceId": 4535},
    {"instanceId": 4686},
    {"instanceId": 46039990},
    {"instanceId": 46039993},
    {"instanceId": 46039997},
    {"instanceId": 460399990},
    {"instanceId": 4603555}
  ],
  "relatingStructure":{
    "instanceId": 135
  }
},
{
  "instanceId": 5007,
  "ifc": "IFCRELCONTAINEDINSPATIALSTRUCTURE",
  "globalId": "2iiyvJC6z28xpx2jnhX4iX",
  "ownerHistory": {
    "instanceId": 41
  },
  "name": null,
  "description": null,
  "relatedElements": [
    {"instanceId": 726},
    {"instanceId": 873},
    {"instanceId": 1531},
    {"instanceId": 45330},
    {"instanceId": 4738},
    {"instanceId": 46039991},
    {"instanceId": 46039992},
    {"instanceId": 46039998},
    {"instanceId": 46039999},
    {"instanceId": 4603666}
  ],
  "relatingStructure":{
    "instanceId": 141
  }
}
```


The resources in Table 42 capture the relationship with each floor of the building. It uses “relatingStructure” property, an attribute of IfcRelContainedInSpatialStructure entity in IFC specification, to provide links to “IfcBuildingStorey” entities specified in “IfcBuildingStorey” sub-collection. Based on the IFC specification, the “IfcBuildingStorey” is specified in “Object Definition” concept, so the “IfcBuildingStorey” sub-collection is defined under “Object Definition” collection and has the URI as: objectdefinitions/ifcproducts/ifcspatialelements/ifcspatialstructureelements/ifcbuildingstoreys.

6.4.3 Received ifcJSON model

As shown in Figure 79, on the client side assuming the authentication has passed and the client is connected to IFC REST API, the user can request for BIM data based on MVD and its concepts and send a GET request to IFC REST API. Upon receipt of data, the MVD modules are visualized in a graph and BIM model is translated to Three JS bindings as explained earlier.

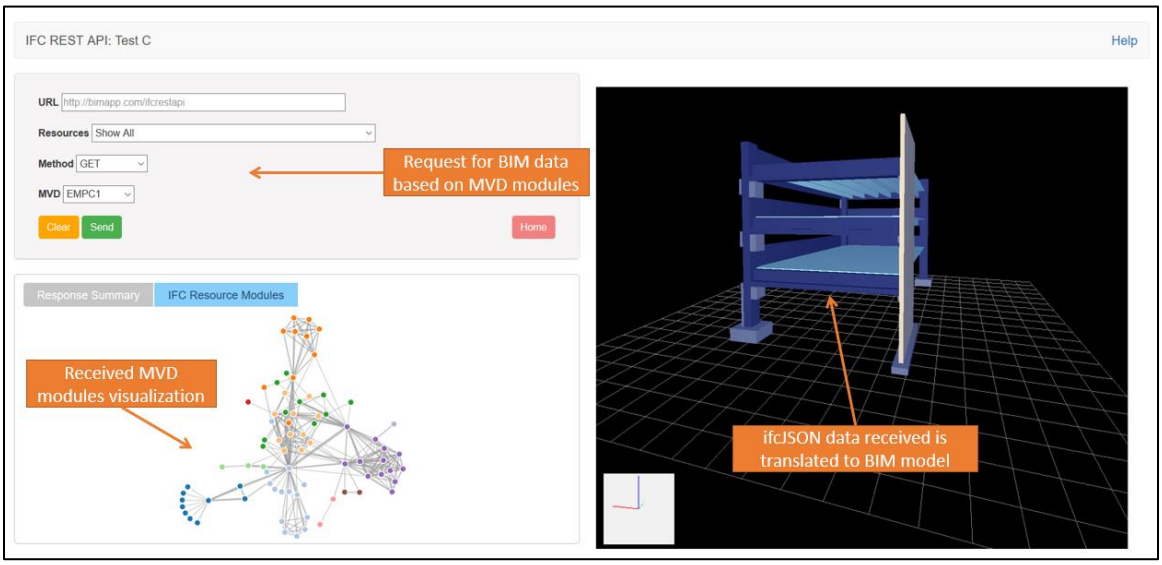


Figure 79 Client side interface

If MVD is set to EMPC.1, under resources as shown in Figure 80, all MVD concepts i.e. REST resources will be listed. Upon data request, all MVD concepts exist in the BIM model will be returned. Alternatively, the user can select a specific collection e.g. ifcbuildingelements to get from the REST API. In this implementation, as mentioned, all

MVD concepts are retrieved at the same time. Using BIM Synapse framework enables data request from the client side based on MVD modules and access to REST API resources.

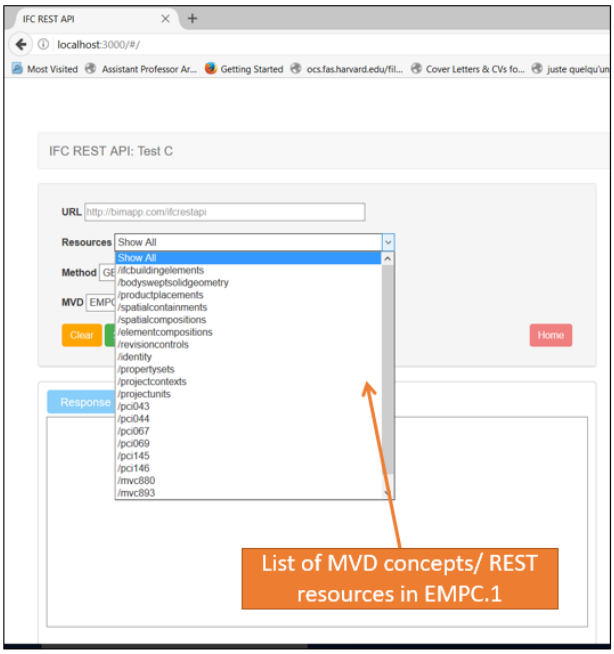


Figure 80 List of available concepts under EMPC.1

6.5 Part 5: Validation and Evaluation

Based on the evaluation metrics explained earlier, the study should validate the data in the designed experiment to evaluate the framework. Thus, the experiment should be validated for the correctness of the BIM model being sent, the accuracy of the framework implementation, and completeness of the received data. In addition, the study should evaluate whether the proposed framework is consistent with what it aimed to achieve and can address the hypotheses of the research. Table 43 shows a summary of the data validation and research evaluation criteria.

Table 43 Summary of the BIM Synapse evaluation methodology

Criteria	What to measure?	How to measure?
Correctness	The data that is being sent as the base BIM model should be a valid data.	To ensure that the data is correct syntactically and semantically the study should perform validation of ifcJSON documents for (1) formatting (2) against the ifcJSON schema.
Accuracy	The implementation of the framework and its components should be accurate.	RESTful IFC API manages BIM data request and response so this can be achieved by using techniques for REST API testing.
Completeness	The received BIM data should be a complete BIM model.	(1) Completeness of the BIM model can be achieved through translating received ifcJSON data back to IFC STEP file and validate the IFC file using available tools. (2) Completeness of modules of data can be achieved by testing modules of received data against data that satisfies modularization in related MVD concepts.
Consistency	The study hypotheses should be met and applied in BIM Synapse.	BIM Synapse framework is supposed to address the study hypotheses so it needs to ensure that the outcome of the procedure can identify the underlying research requirements. This can be achieved by revisiting the results and mapping the evaluation metrics with initial research requirements.

6.5.1 Validating REST Resources

To check the correctness of the BIM model being sent, the ifcJSON resources should be validated. The validation should be both for JSON formatting and for correctness against the schema. As explained in chapter 4, this study uses the "JSON Schema Lint" [80] which is a JSON schema validator that helps with writing and testing JSON data that conform to the JSON schema draft v6 specification.

As explained in chapter 4, for validating REST resources i.e. ifcJSON documents against the ifcJSON4 schema, JSON Schema validator "ajv" is used [81]. This will return all the errors in the ifcJSON collections and assist with generating a correct set of data to be used as REST resources. Previously, REST resources in "IfcBuildingElement" sub-collection were shown in Table 41. These resources have been validated both for formatting and correctness of data against ifcJSON4 schema represented in Table 44. In this implementation, IfcBuildingElement is narrowed down to five of its sub-type classes including IfcColumn, IfcBeam, IfcFooting, IfcSlab, and IfcWall as these are the set of building elements used in the BIM model. Other subtypes such as IfcChimney and IfcWindow are not included because the test model does not carry those instances.

A complete ifcJSON4 schema used in this study is presented in Appendix A which follows both JSON schema and IFC4 specification.

Table 44 ifcJSON schema for IfcBuildingElement collection

REST Collection: IfcBuildingElement

URI: `/api/objectdefinitions/ifcproducts/ifcelements/Ifcbuildingelements`

REST Main Collection: Object Definition

IFC Entity: IfcBuildingElement

ifcJSON Schema


```
{
  "type": "object",
  "properties": {
    "instanceId": {
      "type": "integer"
    },
    "ifc": {
      "type": "string",
      "enum": ["IFCCOLUMN","IFCBEAM","IFCFOOTING","IFCSLAB","IFCWALL"]
    },
    "globalId": {
      "type": "string",
      "maxLength": 22
    },
    "ownerHistory": {
      "oneOf": [
        {"type": "null"},
        {"type": "object",
          "allOf": [
            {"$ref": "#/properties/ifcOwnerHistory"}]
        }]
    },
    "name": {
      "oneOf": [
        {"type": "null"},
        {"type": "string",
          "maxLength": 255
        }]
    },
    "description": {
      "type": ["string","null"]
    },
    "objectType": {
      "oneOf": [
        {"type": "null"},
        {"type": "string",
          "maxLength": 255
        }]
    },
    "objectPlacement": {
      "oneOf": [
        {"type": "null"},
        {"type": "object",
          "allOf": [
            {"$ref": "#/properties/ifcLocalPlacement"}]
        }]
    },
    "representation": {
      "type": "object",
      "allOf": [
        {"$ref": "#/properties/ifcProductDefinitionShape"}]
    },
    "tag": {
      "type": "string",
      "maxLength": 255
    },
    "predefinedType": {
      "oneOf": [
        {"type": "null"},
        {"type": "string",
          "enum": ["COLUMN","PILASTER","USERDEFINED","NOTDEFINED"]
        },
        {"type": "string",
          "enum": ["BEAM","JOIST","HOLLOWCORE","LINTEL","SPANDREL",
          "T_BEAM"]},
```

```
        {"type": "string",
          "enum": ["CAISSON_FOUNDATION","FOOTING_BEAM",
          "PAD_FOOTING","PILE_CAP","STRIP_FOOTING"]},
        {"type": "string",
          "enum": ["FLOOR","ROOF","LANDING","BASESLAB"]
        },
        {"type": "string",
          "enum": ["MOVABLE","PARAPET","PARTITIONING","PLUMBINGWALL",
            "SHEAR","SOLIDWALL","STANDARD","POLYGONAL",
            "ELEMENTEDWALL"]
        }]
    }
  },
  "required": ["ifc","globalId","ownerHistory","name","description",
  "objectType","objectPlacement","representation","tag","predefinedType"]
}
```


6.5.2 Validating IFC REST API

The ifcJSON documents are stored in the persistence layer and exposed as the REST resources to be retrieved through the HTTP calls. For REST API testing, an application needs to interact with the API and involves the calls. Here, since the focus is on retrieving the data, only GET method is used and tested. Steps for testing REST API includes sending a network request to IFC REST API, get a response back, and compare the data against a predetermined result.

To measure the accuracy of the RESTful IFC API this study uses a REST testing solution e.g. SoapUI. SoapUI is an open source cross-platform functional testing solution. With a graphical interface, and specific features. SoapUI allows creating and executing automated functional tests for RESTful web services, their resources and representations while it supports all the standard protocols and technologies [106]. Figure 81 shows the GET request testing when retrieving “IfcBuildingElement” collection in IFC REST API. The raw data response as shown in Figure 82 returns HTTP 200 status i.e. successful HTTP request.

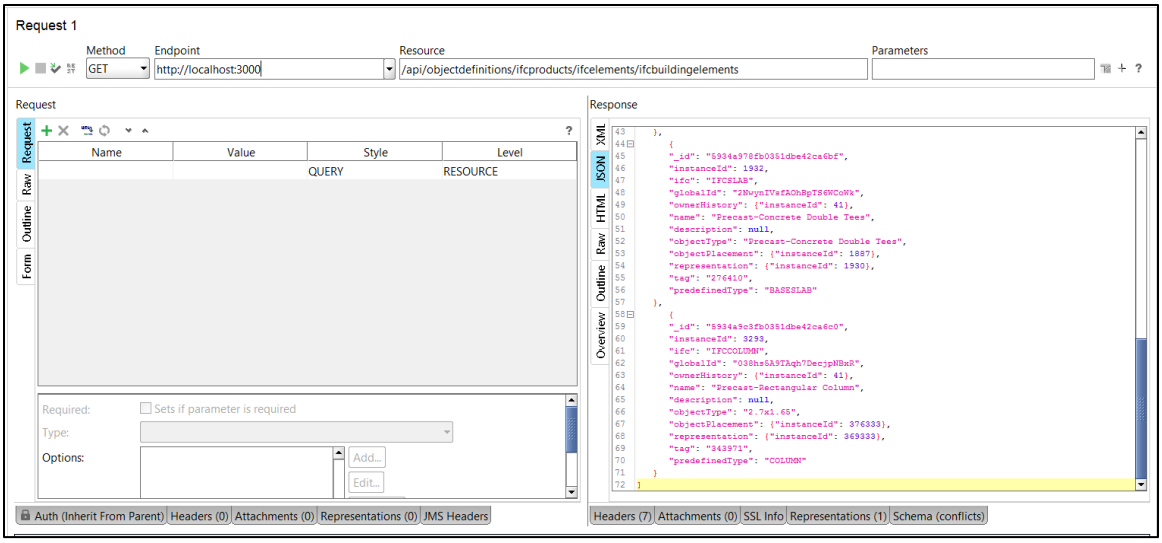


Figure 81 GET request testing for ifcbuildingelements collection

Figure 82 Raw data in REST testing

To get a data and compare the data against a predetermined result, the test uses assertions that compare parts of the data or entire response to some expected value such as

a building element tag, a specific placement data, or a requirement within the spatial containment data. As shown in Figure 83, if the test returns “VALID” message, it means that the corresponding test has passed and fulfills its associated assertion.

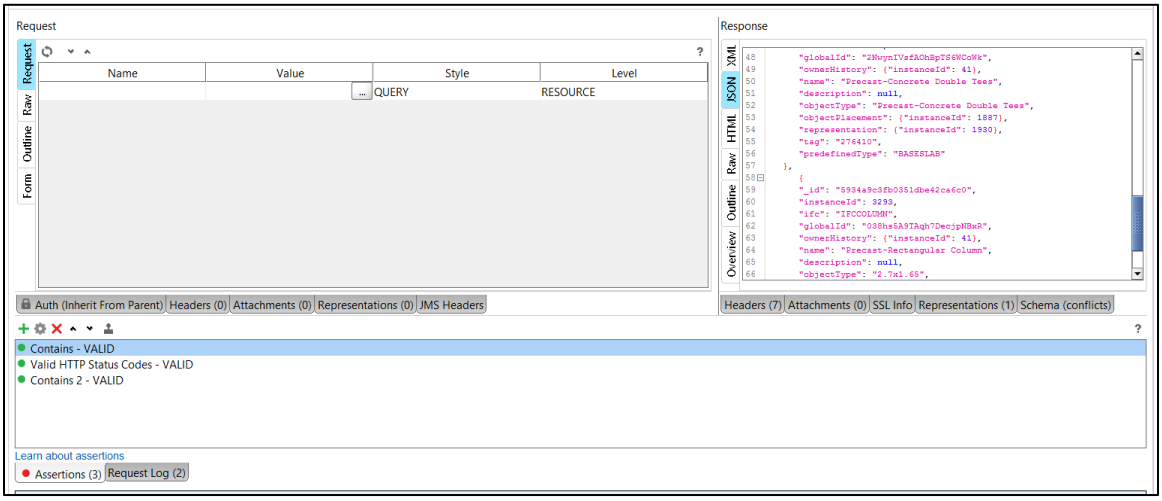


Figure 83 Validating REST data with assertions

Successful tests have ensured the accuracy of the IFC REST API, data within its resources, and execution of HTTP calls.

6.5.3 Validating Received BIM Data

To check the completeness of the received BIM data, there are three major validations. First, REST resources received should be checked with the data sent to compare and ensure that the data is complete. This can be done through REST testing explained above with using assertions. Second, the ifcJSON model can be validated against the ifcJSON schema. Third, all the ifcJSON documents received should represent a complete BIM model. Since there is not currently any tool available to validate the ifcJSON model, the data needs to be translated back to an IFC model and be validated with typical

validation tools like the IfcDoc tool against the PCI model view definition. With these three testing, there is still a chance that some part of data that is received in the client side might be missing but still the BIM model passes all the test. To overcome this challenge, two sets of ifcJSON data in the sender and receiver side should be checked for equality. Figure 84 shows a comparison solution used to compute differences between two ifcJSON documents and reports the list of properties that are different.

```
function filter(res1, res2) {
    var result = {};
    for(key in res1) {
        if(res2[key] != res1[key]) result[key] = res2[key];
        if(typeof res2[key] == 'array' && typeof res1[key] == 'array')
            result[key] = arguments.callee(res1[key], res2[key]);
        if(typeof res2[key] == 'object' && typeof res1[key] == 'object')
            result[key] = arguments.callee(res1[key], res2[key]);
    }
    return result;
}
```

Figure 84 A solution for comparing ifcJSON resources

This process of validation, comparison and analysis indicate that the BIM model received in the client side is a complete BIM model which is the same as the BIM model created initially.

6.5.4 Framework Evaluation

Summary of the data validation methodologies and results discussed so far, are shown in Table 45. The last stage of the evaluation process is to evaluate the study results against the hypotheses set at the beginning and check whether the research outcome is consistent with the study hypotheses. As explained in chapter 1, this research has set three tentative answers to the solution for Cloud-BIM interoperability. Mapping of research hypotheses to study results and data validation methodologies are explained in Table 46.

Table 45 Summary of data validation methodologies and results

Validation	Syntactic Check	Validation against ifcJSON4 Schema	REST testing	Translating received ifcJSON data back to IFC STEP File and Validate the model	Testing modules of received ifcJSON data against original modules
Techniques used	JSON validator jsonlint.com	JSON Schema Validator ajv	SoapUI	IfcDoc tool	Comparison algorithm
Correctness of REST Resources	✓	✓			
Accuracy of IFC REST API			✓		
Completeness of Received BIM Model				✓	✓

First, the study proposed that the IFC specification as BIM open standard can be used to guide a network-based BIM data interoperability in the Cloud. The design and implementation of BIM Synapse have demonstrated how IFC schema and its data model can be used in the underlying components of this framework to guide data exchange using web technologies. The BIM data follows IFC specification and the REST resources are designed based on IFC fundamental concepts. The validation of REST resources has proved the correctness of the BIM data in this framework. BIM Synapse uses the IFC specification to generate the API resources and to arrange them within URIs and modularize the BIM model based on IFC concepts.

Second, the study suggested that Cloud-BIM data interoperability can be achieved with a loosely coupled system that reduces dependencies between sender and receiver of the BIM data. The concept of loosely coupled REST API used in this study has made the Cloud-BIM interoperability possible with minimum dependency on the sending and receiving application. The validation of IFC REST API with REST testing technique has proved the IFC REST API is accurate. In BIM Synapse framework that follows fundamentals of REST architecture, none of the collaborating applications need to know about the components of the other system's architecture. If any of the systems changes its components, there is no need for the framework upgrade if all collaborating applications provide the IFC REST API with required resources that are completely independent of the software components within these systems.

Third, this research argues that the receiver of the BIM model can use MVD specification to request and receive data directly. The URI pattern and data retrieval logic in the IFC REST API has shown how data can be retrieved based on the IFC concepts that corresponds to the MVD specification. In addition, validation of received BIM data and the modules of data response in the framework has proved that the BIM model received over BIM Synapse is a complete BIM model. The case of precast concrete and the EMPC.1 experiment and validation has depicted the methodology that allows for retrieval of BIM data directly on the receiver side with HTTP calls that follow the MVD pattern.

Table 46 Mapping between study hypothesis , study results and data validations

Hypotheses	Results	Validations
(1) IFC specification as BIM open standard can be used to guide a network-based BIM data interoperability in the Cloud.	Design and implementation of BIM Synapse is based on IFC schema. • IFC is used in the underlying components of this framework to guide web-based data exchange. • BIM data being exchanged is based on IFC specification. • REST resources are designed based on IFC fundamental concepts. • URI patterns follow IFC fundamental concepts.	Validation of REST resources (6.5.1) has proved the correctness of the IFC-based data in this framework.
(2) BIM data interoperability can be achieved with a loosely coupled system in the Cloud that reduces dependencies between sender and receiver of model-based BIM data.	Loosely coupled REST API has made the Cloud-BIM interoperability possible with minimum dependency on the sending and receiving application. • In BIM Synapse, none of the collaborating applications need to know about the components of the other system's architecture. If any of the systems changes its components, framework is intact.	Validation of IFC REST API (6.5.2) with REST testing technique has proved the IFC REST API is accurate.
(3) The receiver of the BIM model can use MVD specification to request and receive BIM data directly.	The data retrieval logic in the IFC REST API has followed MVD specification with client interactions.	Validation of received BIM data (6.5.3) and the modules of data response has proved that the BIM model received over BIM Synapse is a complete BIM model.

CHAPTER 7. DISCUSSION AND CONTRIBUTION

This Chapter explains the impact of research and the study contributions to both body of knowledge and the state of practice. This research has addressed current issues of model-based BIM data transfer in the Cloud and has proposed to take advantage of the web technologies to reshape the interoperability of Cloud-BIM applications. Thus, this chapter discusses the framework and the study results in a broader perspective to point out how BIM Synapse can improve Cloud-based collaborations and can guide BIM standardization efforts forward. Also, recommendations for revising IFC specification is discussed.

7.1 BIM Data Representation

This study has indicated the opportunities for Cloud-BIM interoperability and the need for a web compatible data format that follows common agreements and terminologies. As explained, so far IFC specification, as the open standard for AEC industry, is provided as EXPRESS and XSD definitions [56] and IFC data files that are being exchanged between applications use three main data formats including ".ifc" using the STEP physical file structure, ".ifcXML" using the eXtensible Markup Language (XML) document structure, and ".ifcZIP" using the PKzip 2.04g compression algorithm [21]. Besides, XML and JSON are two different data serialization formats used in web applications [107, 47]. These two approaches are applied in data transmission between web applications which typically are the application in Asynchronous JavaScript and XML (AJAX). Since XML and JSON have different features, they have been used in different situations but often only one approach is used in the development to ensure unity and readability [47]. buildingSMART International has implemented the IFC standard using XML technologies

as ifcXML specification [58, 60]. XML is a platform independent language for representing data and has been used in the development of web service applications. However, the performance of web services has shown a significant decrease when using XML data because of the low efficiency of reading and parsing XML data during the execution of services [63]. Based on the measurement of metrics such as the number of objects sent, total time to send the number of objects, average time per object transmission, user CPU utilization, system CPU utilization, and memory utilization, it has been proved that JSON is significantly faster and has higher parsing efficiency than XML [61, 47]. Besides, AJAX has become one of the popular technologies for developing web services [108]. AJAX is a web technology to transfer data between a browser and a server asynchronously and has several advantages over the classic web applications since it reduces response time, server load, and bandwidth of web applications [65]. Initially, XML was used in a wide range of AJAX developments, but gradually XML showed inadequacies because its performance reduces significantly when it is applied to interactive pages [47]. To address the issues of XML-based services, AJAX decreases the server workload by applying JavaScript at the client side. JSON is a native data for JavaScript and this feature makes JSON a proper format to be used for data exchanging in AJAX applications [63]. JSON is a key-value style lightweight data exchange format which is independent of any programming language and unlike XML it is easy for machines to parse and generate while it is also human-readable [47].

This study highlights that there is a need to provide guidelines on how to translate IFC data to JSON to indicate how IFC schema can be represented as JSON specification [109]. Therefore, the study outlines how IFC specification can be represented in JSON

format. The ifcJSON Schema developed in this study, is an alternative to IFC EXPRESS specification and the ifcJSON document is an alternative to the Step Physical File (SPF) representation. In fact, this study addresses how to implement IFC standard as a JSON schema to generate JSON documents for web-based data transfer. The ifcJSON schema and document developed here can facilitate standardization of JSON-based BIM data and can have a major impact on interoperability of web-based BIM applications by unifying BIM data representation based on both industry-wide standards i.e. IFC schema and web compatible data format i.e. JSON.

7.2 Improved Collaborative Process

Current BIM collaboration and data exchange process is mostly based on manual file transfer and in this process, data request by the receiver of data (i.e. receiving application) happens outside of an automated process with no direct control on interacting with required data. Most importantly, there is a major issue in the conventional file export/import process because of current process of data exchange, implements the MVD in the sending application to export the model. While there are methods to validate the model after being exported against the initial exchange requirements of the MVD, there is no methodology or tool to validate the model after being imported. In other words, importing the model in the receiving application will end up with vague and incomplete data and so fat most of the focus has been on the export side than the import.

BIM data exchange might be based on BIM integration solutions which are addressing model federation in a centralized platform/server. BIM collaboration might also follow another type of Cloud integration solutions interconnecting a limited number of

design applications on premise through an interchange hub with the help of plug-ins. These last two processes i.e. Server-based and interchange hub solutions are mainly based on limited data formats and allow a limited number of applications to interoperate. Therefore, collaboration with other applications that are not supported by these systems will again rely on exporting/importing BIM models in the form of files, most likely IFC files. In addition, if any changes applied in one of the collaborating applications, the BIM integration solution, either the server-based or the interchange hub, should be updated to avoid data exchange failure. The Web-based data exchange methodology specified in BIM Synapse suggests redefining the information flow while BIM data exchange is enabled by Web-based data transfer protocols and formats utilizing true potentials of the Cloud technology. It also suggests establishing a loosely coupled integration to reduce dependencies between Cloud applications where BIM data request can happen directly in the receiving application.

Loosely coupled systems can operate independently from one another and can communicate with each other dynamically. In BIM Synapse, the user's browser running a Cloud-BIM application communicates with the web server of another Cloud-BIM application through a REST API using HTTP, without the need to know how the other application works. The advantage of this loosely coupled architecture is that collaborating applications can reside on different Cloud platforms or can update their systems independently and there is no need for the BIM Synapse to be updated and this makes the interoperability robust.

Most importantly, as IoT requires network connectivity and provision of resources in the application layer on top of the network connectivity through WoT, the IFC REST

API makes the resources accessible through standard web-based interactions to facilitate the IoT implementation and integrate IoT with the BIM data. With the use of BIM Synapse framework, Cloud-BIM interoperability will transform to an ecosystem of synapses that will reshape the communication for IoT.

7.3 Interoperability Standard for Cloud-BIM

Standards provide a common language and set of expectations that enable interoperability between systems. Similarly, data exchange schema and interoperability standards allow data to be shared between applications regardless of the application or application vendor. Some existing standards can assist with Cloud interoperability on a technical level but to address semantic and organizational interoperability, domain knowledge is required and therefore, established schema and use cases enable the interoperability. Therefore, domain specific standards are crucial for Cloud interoperability. BIM Synapse framework with its IFC REST API and data structure that follows IFC specification, provides a common understanding of BIM data, resource management, and data request/response in the Cloud. BIM Synapse also integrated MVD as a subset of IFC schema to address specific exchange requirements. BIM Synapse is among the first efforts towards standardization of the AEC data in the Cloud. Cloud-BIM interoperability requires industry engagement to establish the standard and certify BIM applications. It needs the involvement of BIM software vendors to implement the framework and conduct user testing.

Since BIM Synapse uses ifcJSON as its main data encoding, it has the benefit of explicit semantics over conventional SPF data where the semantics are hidden. To interpret

the data in an SPF file, the IFC specification is needed to provide an understanding of the attributes. But ifcJSON can capture, store, and aggregates the semantics of data and provide the mapping with explicit representation of attributes or properties. This can improve exchange semantics when dealing with IFC specification. In addition, the design of IFC REST API has shown how the resources can be linked in BIM Synapse. Therefore, the use of ifcJSON and its implementation for REST resources can improve the semantic uniformity and can address the needs for linked data. Besides, JSON-LD has been introduced to standardize linked data in JSON. It is compatible with JSON and is considered as a first step for standardizing semantic RESTful web services. The ifcJSON-LD schema as a future step can improve linked data functionality.

It is anticipated that ifcJSON will be of interest to Web application development and Cloud computing community to replace ifcXML in most cases. Therefore, the ifcJSON implementation as well as REST resource management described in this study, needs to be considered for the full generation of the whole IFC schema to be added as a buildingSMART standard. In addition, this study recommends developing a methodology to generate ifcJSON from IFC EXPRESS automatically considering the methodology described in this research. Also, an automated translation process for mapping between IFC SPF file and ifcJSON document would facilitate the translation of IFC data from EXPRESS to JSON. The AEC software market indicates that Building Information Modeling has been moving to the Cloud with the emergence of BIM web-based applications and BIM server technologies. Thus, the need for JSON-based data transfer will evolve and it would be useful if application developers develop methodologies to

convert their native bindings to ifcJSON directly without the need to use IFC EXPRESS data.

7.4 Required Revisions to IFC Specification

This study has looked at IFC specification to enable its implementation for the benefit of Cloud-BIM interoperability. BIM Synapse framework proposed in this study used IFC data model heavily to guide resource creation and management within its IFC REST API. The detailed study of IFC specification and its fundamental concepts has pointed out some aspects that need to be changed in the IFC specification to improve the data schema and to improve its usability for the Cloud-based data exchange.

The first aspect to be highlighted is the need for unification of URI naming. This research has proposed to use IFC concepts and its naming convention for resource URIs. Since IFC concepts have not been generated with the use of URI patterns in mind, the naming does not express a short and quick pattern for URIs as required. Also, there is a need to revise some of the IFC concepts as some concepts can be merged or summarized and some new concepts need to be added for the better use of IFC concepts for REST resources. For example, a separate concept is needed for Object Definition. Although "Object Definition" concept is specified separately, there is no explicit definition of the concept in IFC4 documentation. Concept definition for "Object Definition" in IFC specification is missing and it repeats the IfcObject entity for any related object. As a result, IfcDoc MVD testing repeats for every concept that duplicates the ifcObject and if anything is wrong with the object all concepts return an error. Although this is an ifcDoc

implementation issue, because IFC is heavily relying on ifcDoc for its documentation, this implementation mistake should be considered.

Another example is the “Project Context” concept shown in Figure 85. The definition of “Project Context” concept in IFC specification specifies the IfcContext as the starting entity of the concept emphasizing that the concept that deals with IfcGeometricRepresentationContext should be applied to IfcContext like an IfcProject entity. While this specification is true, it is not complete because the IfcGeometricRepresentationContext is also used in all the IfcShapeRepresenatation entities. In fact, all “Body Geometry” concepts such as “Body SweptSolid Geometry”, shown in Figure 24 share the IfcGeometricRepresentationContext. Therefore, the concept needs to be defined in a way to capture this relationship with IfcGeometricRepresentationContext in addition to IfcContext relationship.

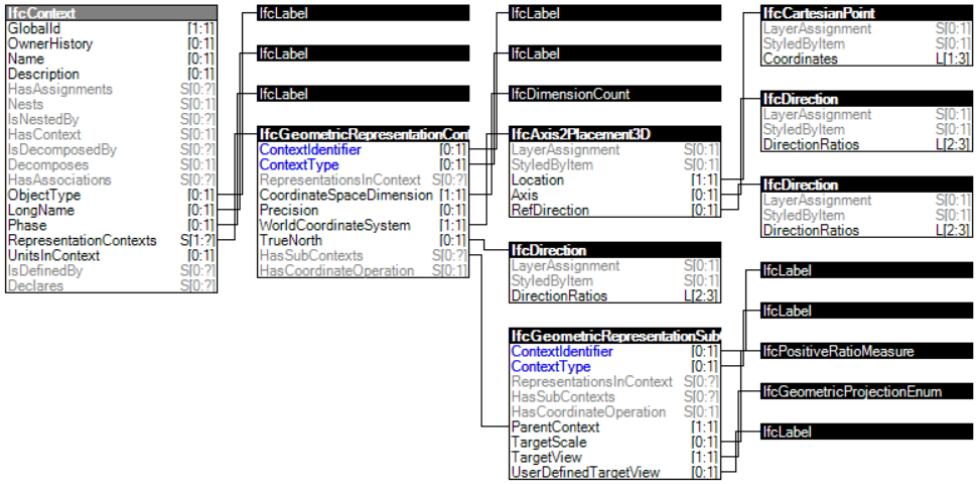


Figure 85 Project Context concept diagram

The second aspect regarding IFC modification is the presence of Some redundant data in the IFC schema. For example, the IfcTelecomAddress includes some attributes like pager number that is not the case anymore. Also, it defined "UserDefinedPurpose" as well as "Purpose" attribute which involves redundant data. Also, some entities are duplicated in multiple IFC fundamental concepts making the concept definitions vague.

The third aspect is the distinction between IFC concepts from rule sets. Some IFC concepts are more of a rule set than a concept definition. For example, "Identity" concept in IFC specification only takes care of Global ID attribute which can be set as a rule and there is no need for a separate concept to address that. In the MVD for precast concrete some concepts are applying rules than modularization. For example, MVC-895 can be added as a rule set to "Spatial Containment" concept in IFC.

Overall, with REST resource identification and URI pattern in mind, there is a need to revise the IFC schema and its concepts. It needs to redefine the IFC specification based on the needs of web technologies standardization.

7.5 Contribution

This research has addressed the issue of interoperability for BIM data exchange by providing retrievable resources, storage layers, and interfaces, which are based on the Cloud technologies as well as IFC open standard. This provides the possibility to use the BIM Synapse framework in different scenarios to share and distribute data back and forth between Cloud-BIM applications in real-time. By using IFC schema and Cloud standards for BIM data exchange, this study helps to move the work on BIM standardization forward

and assist with delivering an architectural model and uses case input for the next generation of the Cloud-BIM Standard.

7.5.1 Contribution to the Body of Knowledge

The major contribution of this study is to assist with standardization of Cloud-based BIM data exchange by defining an interoperability framework based on industry open standard for Cloud-BIM applications to address model-based data exchange. Additionally, it has been pointed out by NIST that for developments in the areas of Cloud standards, it is more likely that data formats will be standardized instead of having Cloud application interfaces integrate [1]. Thus, another contribution of this research is the implementation of ifcJSON data serialization that adopts both IFC schema and JSON schema to outline a systematic encoding of IFC data for Cloud-BIM. In BIM Synapse framework, the work on data encoding, REST resource identification, URI pattern specification, and the MVD integration will contribute to the standardization of AEC data and collaboration process in the Cloud.

Another contribution of this study is that it redefines and re-structures the dataflow for building data exchange in the Cloud. This dataflow addresses current challenges of exporting and importing IFC models that do not include proper model validation technique so far. The proposed dataflow gives the receiver of data more control on requesting data directly.

Moreover, the method of framework evaluation explained in the previous chapter, is an important contribution. As explained in the work of [5], very few of the existing Cloud-BIM research has evaluated the proposed solutions and there is a lack of detail

studies about the evaluation methods for Cloud-BIM and how the evaluations can be conducted. Therefore, the methodology used for the evaluation of BIM Synapse is considered as another contribution of this work.

In addition, this framework is based on web technologies, so it contributes to deploying web technologies in the BIM process. It changes the data exchange conventions from a file-based process to a web-based data transmission. This allows building data to be part of the World Wide Web and to be connected over the networks. Thus, BIM Synapse framework can help incorporating IoT into buildings and within the BIM process. As the data in this framework is web compatible and can be transferred over the web protocols, the framework can enable the integration of IoT with all data available over a network. This could bridge the gap between the data that reside on the BIM models and the data that is required for the Internet of Things (IoT) such as safety, predictive modeling and planning in design, and implementation of smart buildings.

7.5.2 Contribution to the State of Practice

BIM applications are gradually moving to the Cloud. The lack of standardization especially in developing the Cloud APIs will end up with many challenges for the systems to interoperate. BIM Synapse framework can play a significant role among Cloud service providers and the framework implementation can be used as a benchmark towards implementing interoperable Cloud-BIM applications that are designed based on loosely coupled collaboration. The IFC REST API can either guide the generation of Cloud-BIM APIs or can be added as an API layer to existing systems to extend the capabilities of Cloud-BIM applications.

Moreover, in the evaluation of BIM Synapse, the study has revised PCI MVD concepts and therefore, it has contributed to precast concrete domain and precast concrete BIM standard by upgrading and revising the MVD concepts.

Besides, in the AEC industry with a growing range of tools and applications embedding the BIM Synapse in the BIM workflow can assist the project parties with smoother and faster collaborative process. This can expedite the production and revision of design and construction documents. As BIM Synapse helps to provide direct access to the BIM data in the collaborating Cloud-BIM application, it improves the visibility of data during the project development and helps with decision-making process.

CHAPTER 8. CONCLUSION

This Chapter summarizes the results of this research and reviews the research questions. Study limitations and its major findings are explained.

8.1 Challenges and Limitations

In this research, as explained in Chapter 5 section 5.5.5, since the focus is on BIM data transfer, only the GET operation of HTTP is used. The HTTP GET method can read and retrieve a representation of a resource i.e. REST resource. Standard HTTP operations are GET, POST, PUT and DELETE and since BIM Synapse in this research emphasizes on data retrieval, this study does not implement POST, PUT, or DELETE operations.

Moreover, the implementation of BIM Synapse in this study is limited to precast concrete MVD and the exchange requirement for EMPC.1 as the conceptual model of precast concrete. It is limited to 20 main MVD concepts within EMPC.1 including 11 IFC4 fundamental concepts and 9 PCI MVD concepts. These selected set of concepts include precast elements with their geometric representation in the BIM model with required aggregation both in the element level and project's spatial hierarchy. They also carry PCI specific data regarding projection elements and spatial elements requirements. Data regarding precast object association, element type, group assignment, joint and surface treatment are excluded.

Besides, similar to the limitations of ifcXML Schema [58], the ifcJSON Schema in this study does not contain "Inverse" relationships and "derived attributes" that an EXPRESS schema can include. However, the implementation of JSON "$ref" keyword -

as explained in Chapter 5, Section 5.5.4 and in Chapter 6, Section 6.4.2- can ensure the "Inverse" relationships in practice. In addition, ifcJSON document does not include the "WHERE" rules that in EXPRESS are added to entities to restrict the range or combinations of attribute values. As suggested in ifcXML specification [58], this study would also recommend the applications generating ifcJSON to ensure these constraints are met in the implementation.

Another limitation of this study which is the limitation of ifcJSON4 implementation is regarding the uniqueness of GUIDs and instanceIds. Generally, it is not possible to check the uniqueness of properties such as instanceIds and GUIDs in JSON schema. JSON is quite limited for making data values validation because it is not the purpose of the standard. This research recommends that the functionality of checking the uniqueness of values should be implemented as an added function in JavaScript code or an array of IDs.

Additionally, there is currently no tools available that can visualize geometry data in ifcJSON. Therefore, for visualization purposes this study has used Three JS as a client side viewer running in the browser. The mapping between ifcJSON documents and Three JS geometry has been implemented in an ad-hoc process and providing the mapping guidelines is not in the scope of this study. The mapping from ifcJSON to any native binding should be done case by case but it can be easily managed as the ifcJSON resources have made the modularization possible.

For validation of IFC data and MVD related testing, this study uses IfcDoc tool. The experience has shown that IfcDoc tool implementation has limitations and might return

a false positive validation report. Therefore, the process has used manual checking, testing, and comparisons at times.

As mentioned, the issues regarding security, ownership and stability for Cloud-BIM applications are open research areas but those issues are not in the scope of this research.

8.2 Concluding Remarks

Cloud-BIM is known as the second generation of BIM development and interoperability is key to the successful implementation of BIM. This study emphasized that currently there is a gap in research regarding the identification of technologies that can assist with Cloud-BIM interoperability solutions to achieve a loosely coupled network-based data exchange process. The research goal has been to address five major questions:

1. **What are the current limitations of existing Cloud-based interoperability for BIM collaborations in Cloud?**

   The study has indicated in chapter 2 that Cloud-BIM data transmission for cross-platform collaboration faces several challenges including standardization and data interdependency. It showed that existing architectures for Cloud-BIM data integration face several challenges in model-based data exchange and have not fully exploited the potential of the Cloud.

   Current Cloud-BIM interoperability approaches can be categorized to three types of dataflow architecture. First, manual file transfer that is currently a common way of exchanging BIM data across applications. Project data can be exported and shared in the form of vendor-specific formats or neutral format using IFC standard. Second, BIM server technologies in which server-based BIM solutions known as model collaboration systems have provided a central BIM service with a single-sourced data server

accessible for project partners. These model server technologies utilize information directly from the models and are intended to improve multidisciplinary collaboration. The issue with BIM server technologies is that their implementations are still limited thus have faced scalability issues. Third, Data Interchange Hub such as Flux project that can automate dataflow between certain applications. This type of solution currently has a very limited implementation and supports very few design applications to be connected via the hub. Moreover, in this approach the inter-connection of applications relies on the hub solution and its capabilities although supported software packages can exchange data directly. The study indicates that these existing Cloud-BIM interoperability solutions face and mix different challenges and have not fully utilized the potential of Cloud computing by implementing a standardized network-based data transmission process.

The challenge highlighted in this study is regarding interoperability architecture for an improved collaborative process in the Cloud. There is a lack of a framework to redefine the dataflow in Cloud-BIM data exchange process while utilizing web-based technologies as major enablers of the Cloud.

2. What approaches and techniques of web technologies can assist with BIM data transmission to address a network-based data exchange in the Cloud?

The study explained in chapter 3 the opportunities in web technologies that can assist Cloud-BIM interoperability with alternative approaches and techniques. Major features of Cloud interoperability such as APIs, data transfer protocols, data formats and standardization efforts were discussed. This study highlighted that current cross-platform Cloud-BIM data transfer methodologies which are mainly based on manual file transfer, data conversion and data merge, have so far neglected to deply these foundations of web architecture. Besides, collaboration and data sharing among

multiple Cloud applications are based on the type of the collaborative environment including federated collaboration, loosely coupled collaboration, and ad hoc collaboration. Among all, the architecture for loosely coupled collaboration reduces dependencies between Cloud applications as well as components of the collaborating applications. So, data exchange can be significantly simplified. Therefore, this study suggests the architecture for loosely coupled collaboration to support Cloud-BIM data interoperability.

The study of Cloud integration solutions being used in other domains than AEC indicated that current solutions allow the integration of two or multiple application to a third new system and they do not allow for a loosely coupled collaboration. These Cloud interoperability solutions suffer from vendor lock-in and several problems with resource management and data sharing because each system uses their own interoperability framework and data schema. Data integration must expect data to conform to a common schema to address interoperability challenges.

3. What dataflow architecture should be applied for BIM data exchange in the Cloud to address current challenges?

The study showed in chapter 3 and 5 how major components of Cloud interoperability can be deployed to improve BIM data transmission and address a network-based data exchange for a collaborative workflow. The study proposed new dataflow and a framework as BIM Synapse. In this architecture, data for Cloud collaboration is stored in a data-store within sending application and can be accessed through an API with a standard set of methods and terminologies. The data can be interpreted by a collaborating application which is the receiving application. The API in fact, provides an on-demand access to the pool of data on sending application. Accordingly, data request happens directly in the receiving application that uses a set

of data exchange requirements within MVD specification to request for the BIM data. The data exchange requirements in this process follow the rules defined in the MVD specification.

The dataflow introduced in BIM Synapse framework can address current challenges of BIM data exchange in the Cloud by providing a loosely coupled network-based data interoperability approach. This research specified the dataflow for Cloud-BIM within BIM Synapse to utilize true potentials of the web technologies. The proposed dataflow can perform as an enabler for a collaborative process allowing Cloud-BIM services communicate through their standardized APIs.

4. **How can IFC specification - as the open BIM standard in the AEC industry- enable data interoperability in Cloud-BIM applications?**

Cloud-BIM interoperability research and its standardization effort is still in its infancy, and the body of knowledge in the area has not been well defined yet. Therefore, this study highlighted in chapter 4 and 5 the need for standardization of Cloud-BIM APIs using IFC data model as industry established schema. IFC specification can assist with semantic interoperability of building data in the Cloud. This research has investigated how the IFC data model can be integrated in the standardization of Cloud-BIM APIs with the implementation of a RESTful IFC API. IFC REST API integrates IFC specification within the identification of the API resources, the specification of the URI patterns, and encoding of web compatible data.

The study highlighted that while vendor-specific data formats are quite diverse, these are based on multiple and different data schemas. At the same time, data standards for Cloud-based cross-platform data exchange purposes are limited. IFC data model which describes building data provides a means to define building components and processes in a publicly available data schema. As an open standard for the AEC

industry, IFC schema definition has not become the basis for Cloud-BIM integration solutions although it can ensure a common understanding of building data across the applications and disciplines. IFC encoding is currently only certified for STEP file and XML for encoding documents. The importance of JSON data for replacing XML documents to improve the performance of web-based data transfer has been explained and proved in several studies. This study introduced ifcJSON and demonstrated that IFC Schema should and can be implemented in JSON format. JSON as a lightweight data exchange format has been proved to have higher parsing efficiency than XML and has been successful in replacing XML in JavaScript-based web applications. Therefore, since JSON representation of IFC data is required as an alternative to ifcXML, the ifcJSON4 is developed which is the JSON implementation of the current release of IFC schema that is IFC4 Add2.

This research developed a methodology to generate ifcJSON schema that is constrained by the JSON schema as well as IFC specification (i.e. IFC EXPRESS Schema). Then, in its implementation for a use case, the study explained in more details how the data content in ifcJSON document can be generated. Most importantly, the study demonstrated that ifcJSON4 schema is a well-formed and valid JSON schema and the ifcJSON documents that can be generated based on ifcJSON4 schema are valid JSON documents. Despite the challenges and limitations of ifcJSON implementation, the methodology and implementation of ifcJSON4 described here shows that IFC can be represented in JSON format to be applied towards web-based data exchange in the AEC industry replacing ifcXML. Data exchange is a very important issue for making Cloud computing more efficient and JSON data format supports high scalability. Therefore, ifcJSON developed in this study is anticipated to be widely used in Cloud-based Building Information Modeling solutions to improve interoperability of Cloud-BIM applications.

Without a doubt, there is a need for standardization of building data on web and this study indicated how such standardization can be applied with the introduction of ifcJSON. This standardization effort can be done by using IFC schema as the industry established open standard and JSON as an open standard and language-independent data format. Unless ifcJSON schema is completely established, the data on the web will lean towards proprietary schemas with no common structure or standard schema to translate between vendor-specific data contents. The methodology for developing ifcJSON schema and document described in this research can facilitate standardization of JSON-based BIM data. As a matter of fact, standardization of web compatible BIM data can expedite the production and revision of construction documents.

BIM Synapse framework with its IFC REST API and data structure follows IFC specification and provides a common understanding of BIM data, resource management, and data request/response in the Cloud. Its RESTful IFC API is based on a set of REST API design patterns and uses ifcJSON for resource representation. This makes the resources compatible with REST as it follows JSON data model and additionally, it aligns the resources with AEC standards since it follows IFC specification. The identification of API resources and the resource schema itself is based on IFC concepts and MVDs. Thus, the API resources are standardized for the AEC industry while it also captures domain semantics since MVD efforts are aimed at providing domain-specific semantic clarity for BIM exchanges. BIM Synapse integrates MVD as a subset of IFC schema to address specific exchange requirements. In fact, in BIM Synapse framework, the ifcJSON data encoding, REST resource identification, URI pattern specification, and the MVD integration follow the IFC specification.

5. **How can MVD- as the information exchange standard- facilitate BIM data exchange in the Cloud?**

Chapter 5 and 6 has explained that in BIM Synapse framework proposed in this study, the client application can access the API and retrieve the BIM data from the server application based on HTTP calls that retrieve the BIM model based on MVD modularization. In this architectural model, the MVD implementation will be an effort in the receiving application. The study also emphasized the need for revision to MVD specification such as in precast concrete Model View Definition. The MVD should use IFC common concepts and add specialized concepts based on the needs of the domain. In this study, MVD for precast concrete has been revised so that it can be upgraded to IFC4 specification and integrated IFC4 fundamental concepts with MVD specific concepts.

In BIM Synapse framework, the user can request for BIM data in the client side based on MVD and its concepts to send a GET request to IFC REST API. Each MVD contains a set of concepts and requesting for a BIM model based on an MVD set of concepts will return all ifcJSON resources that is included in the MVD.

Table 47 Key characteristics of BIM Synapse and other systems

Solutions	Scalable	Loosely coupled	Can capture domain knowledge	Network-based data exchange	Standard-based	Use of Web technologies	WoT & IoT support
BIM Synapse							
Cloud Integration Solutions							
Data Interchange Hub							
BIM Server Technologies							
Manual File Transfer							

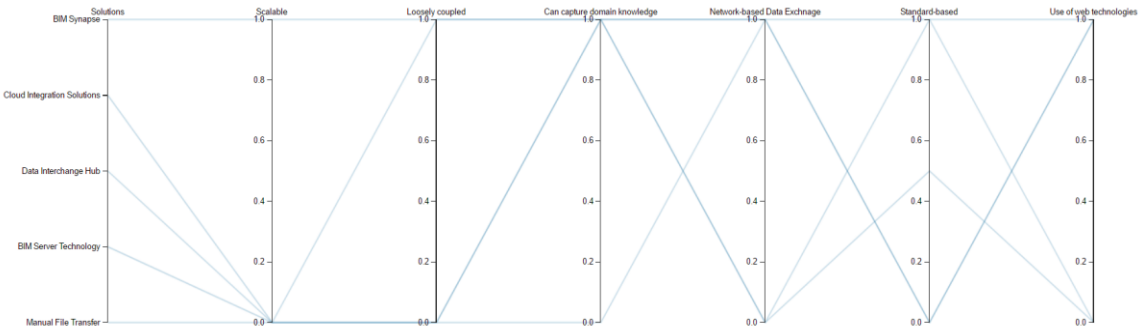


Figure 86 Key characteristics of BIM Synapse and other systems

Overall, BIM Synapse framework proposed in this research, can address challenges of current BIM interoperability in the Cloud by using web technologies and providing a network-based dataflow architecture. Figure 86 and Table 47 illustrate a matrix with key characteristics of BIM Synapse in comparison with other solutions such as file-based data transfer, BIM server technologies, data interchange hub technologies, and existing Cloud integration solutions studied in this research in chapter 2 and 3. RESTful IFC API in the BIM Synapse framework as well as the NoSQL database that persists the BIM data and JSON representation of resources all provide the framework with high scalability and simplicity. Also BIM Synapse creates a loosely-coupled collaboration between Cloud-BIM applications. This framework emphasizes on a loosely coupled collaboration and uses standards to design, implement and evaluate the framework. BIM Synapse can capture domain knowledge by using MVD concepts to retrieve REST resources. It uses HTTP as the main data transfer protocol and creates a network-based data transmission. BIM Synapse uses existing web technologies and emphasizes on the use of standards as the underlying basis of the data exchange. If Cloud-BIM services are designed and

implemented with no standardization in mind they are subject to the same data interoperability problems as current challenges with desktop-based BIM applications.

Moreover, BIM Synapse can enable the integration of IoT with BIM model. As IoT requires network connectivity and provision of resources in the application layer on top of the network connectivity through WoT, the IFC REST API in BIM Synapse framework makes the resources accessible through standard web-based interactions. Thus, it enables the integration of IoT with the BIM data. Implementation of BIM Synapse framework will transform Cloud-BIM interoperability to an ecosystem of synapses as shown in Figure 87, that will reshape the collaboration in the AEC industry and will enable the communication for IoT.

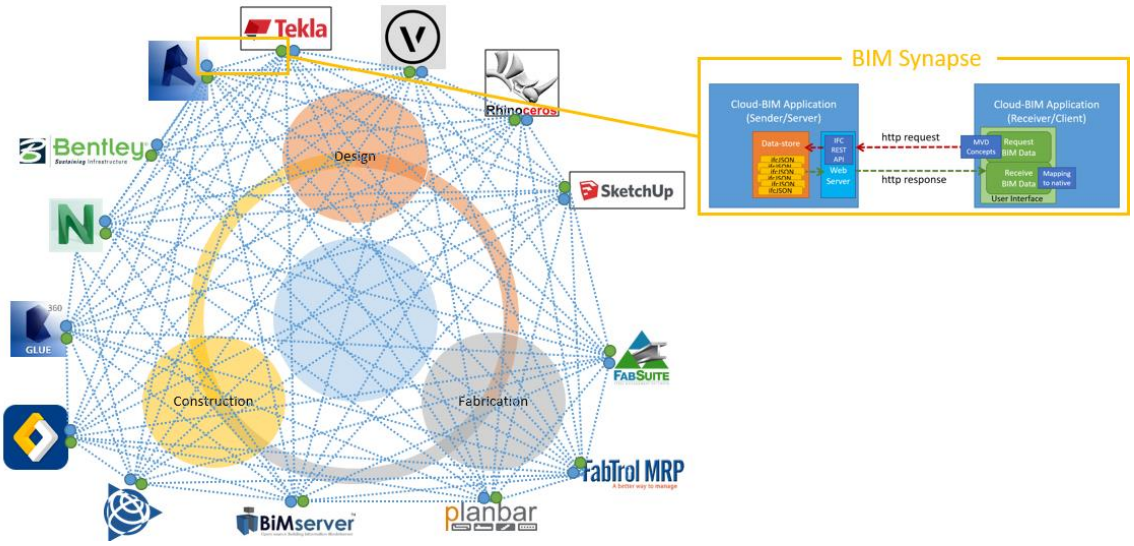


Figure 87 An ecosystem of the BIM Synapses during the project lifecycle

APPENDIX A. SCHEMA

This appendix provides ifcJSON4 schema developed and used in this study.

ifcJSON4 Schema

```
{
	"$schema": "http://json-schema.org/draft-04/schema#",
	"title": "ifcJSON4 Schema",
	"description": "This is the schema for representing IFC4 data in JSON",
	"definitions": {
		"description": {
			"_comment": "This section includes IFC datatypes."
		},
		"ifcUnit": {
			"type": "object",
			"oneOf": [{
					"$ref": "#/properties/ifcDerivedUnit"
				},
				{
					"$ref": "#/properties/ifcNamedUnit"
				},
				{
					"$ref": "#/properties/ifcMonetaryUnit"
				}
			]
		},
		"ifcAxis2Placement": {
			"type": "object",
			"oneOf": [{
					"$ref": "#/properties/ifcAxis2Placement2D"
				},
				{
					"$ref": "#/properties/ifcAxis2Placement3D"
				}
			]
		}
	},
	"type": "object",
	"properties": {
		"description": {
			"_comment": "This section includes IFC entities"
		},
		"ifcCartesianPoint": {
			"type": "object",
			"properties": {
				"instanceId": {
					"type": "integer"
				},
				"ifc": {
					"type": "string"
				},
				"coordinates": {
					"type": "array",
					"items": {
						"type": "number"
					},
					"minItems": 1,
```

```
                                    "maxItems": 3
                             }
                      },
                      "required": ["ifc", "coordinates"]
               },
               "ifcAxis2Placement2D": {
                      "type": "object",
                      "properties": {
                             "instanceId": {
                                    "type": "integer"
                             },
                             "ifc": {
                                    "type": "string"
                             },
                             "location": {
                                    "type": "object",
                                    "allOf": [{
                                           "$ref":
"#/properties/ifcCartesianPoint"
                                    }]
                             },
                             "refDirection": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "object",
                                                  "allOf": [{
                                                        "$ref":
"#/properties/ifcDirection"
                                                  }]
                                           }
                                    ]
                             }
                      },
                      "required": ["ifc", "location", "refDirection"]
               },
               "ifcAxis2Placement3D": {
                      "type": "object",
                      "properties": {
                             "instanceId": {
                                    "type": "integer"
                             },
                             "ifc": {
                                    "type": "string"
                             },
                             "location": {
                                    "type": "object",
                                    "allOf": [{
                                           "$ref":
"#/properties/ifcCartesianPoint"
                                    }]
                             },
                             "axis": {
                                    "type": ["object", "null"]
                             },
                             "refDirection": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "object",
                                                  "allOf": [{
```

```
                                                        "$ref":
"#/properties/ifcDirection"
                                                }]
                                        }
                                ]
                        }
                },
                "required": ["ifc", "location", "axis", "refDirection"]
        },
        "ifcDirection": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "directionRatios": {
                                "type": "array",
                                "items": {
                                        "type": "number"
                                },
                                "minItems": 2,
                                "maxItems": 3
                        }
                },
                "required": ["ifc", "directionRatios"]
        },
        "ifcRectangleProfileDef": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "profileType": {
                                "type": "string",
                                "enum": ["AREA", "CURVE"]
                        },
                        "profileName": {
                                "type": ["string", "null"]
                        },
                        "position": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcAxis2Placement2D"
                                                }]
                                        }
                                ]
                        },
                        "xDim": {
                                "type": "number",
                                "minimum": 0,
                                "exclusiveMinimum": true
                        },
```

```
                        "yDim": {
                                "type": "number",
                                "minimum": 0,
                                "exclusiveMinimum": true
                        }
                },
                "required":    ["ifc",    "profileType",    "profileName",
"position", "xDim", "yDim"]
        },
        "ifcExtrudedAreaSolid": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "sweptArea": {
                                "type": "object",
                                "allOf": [{
                                        "$ref":
"#/properties/ifcRectangleProfileDef"
                                }]
                        },
                        "position": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcAxis2Placement3D"
                                                }]
                                        }
                                ]
                        },
                        "extrudedDirection": {
                                "type": "object",
                                "allOf": [{
                                        "$ref": "#/properties/ifcDirection"
                                }]
                        },
                        "depth": {
                                "type": "number",
                                "minimum": 0,
                                "exclusiveMinimum": false
                        }
                },
                "required":     ["ifc",     "sweptArea",     "position",
"extrudedDirection", "depth"]
        },
        "ifcGeometricRepresentationContext": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "contextIdentifier": {
```

```
                                    "type": ["string", "null"]
                             },
                             "contextType": {
                                    "type": ["string", "null"]
                             },
                             "coordinateSpaceDimension": {
                                    "type": "integer",
                                    "minimum": 0,
                                    "maximum": 3,
                                    "exclusiveMinimum": false
                             },
                             "precision": {
                                    "type": ["number", "null"]
                             },
                             "worldCoordinateSystem": {
                                    "type": "object",
                                    "allOf": [{
                                           "$ref":
"#/properties/ifcAxis2Placement3D"
                                    }]

                             },
                             "trueNorth": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "object",
                                                  "allOf": [{
                                                        "$ref":
"#/properties/ifcDirection"
                                                  }]
                                           }
                                    ]
                             }
                      },
                      "required":  ["ifc",  "contextIdentifier",  "contextType",
"coordinateSpaceDimension", "precision", "worldCoordinateSystem", "trueNorth"]
               },
               "ifcShapeRepresentation": {
                      "type": "object",
                      "properties": {
                             "instanceId": {
                                    "type": "integer"
                             },
                             "ifc": {
                                    "type": "string"
                             },
                             "contextOfItems": {
                                    "type": "object",
                                    "allOf": [{
                                           "$ref":
"#/properties/ifcGeometricRepresentationContext"
                                    }]
                             },
                             "representationIdentifier": {
                                    "type": ["string", "null"]
                             },
                             "representationType": {
                                    "type": ["string", "null"]
                             },
                             "items": {
                                    "type": "array",
```

```
                                    "items": {
                                          "type": "object",
                                          "allOf": [{
                                                "$ref":
"#/properties/ifcExtrudedAreaSolid"
                                          }]
                                    },
                                    "minItems": 1
                              }
                        },
                        "required":           ["ifc",           "contextOfItems",
"representationIdentifier", "representationType", "items"]
                  },
                  "ifcProductDefinitionShape": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "name": {
                                    "oneOf": [{
                                          "type": "null"
                                    },
                                    {
                                          "type": "string",
                                          "maxLength": 255
                                    }
                                    ]
                              },
                              "description": {
                                    "type": ["string", "null"]
                              },
                              "representations": {
                                    "type": "array",
                                    "items": {
                                          "type": "object",
                                          "allOf": [{
                                                "$ref":
"#/properties/ifcShapeRepresentation"
                                          }]
                                    },
                                    "minItems": 1
                              }
                        },
                        "required":      ["ifc",      "name",      "description",
"representations"]
                  },
                  "ifcLocalPlacement": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "placementRelTo": {
                                    "type": "null"
                              },
                              "relativePlacement": {
```

```
                                    "type": "object",
                                    "allOf": [{
                                          "$ref":
"#/properties/ifcAxis2Placement3D"
                                    }]
                              }
                        },
                        "required": ["ifc", "placementRelTo", "relativePlacement"]
                  },
                  "ifcPerson": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "identification": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
                                    ]
                              },
                              "familyName": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
                                    ]
                              },
                              "givenName": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
                                    ]
                              },
                              "middleNames": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "array",
                                                "items": {
                                                      "type": "string",
                                                      "maxLength": 255
                                                },
                                                "minItems": 1
                                          }
                                    ]
                              },
                              "prefixTitles": {
```

```
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "array",
                                                "items": {
                                                      "type": "string",
                                                      "maxLength": 255
                                                },
                                                "minItems": 1
                                          }
                                    ]
                              },
                              "suffixTitles": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "array",
                                                "items": {
                                                      "type": "string",
                                                      "maxLength": 255
                                                },
                                                "minItems": 1
                                          }
                                    ]
                              },
                              "roles": {
                                    "type": "null"
                              },
                              "addresses": {
                                    "type": "null"
                              }
                        },
                        "required":   ["ifc",   "identification",   "familyName",
"givenName",   "middleNames",   "prefixTitles",   "suffixTitles",   "roles",
"addresses"]
                  },
                  "ifcPersonAndOrganization": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "thePerson": {
                                    "type": "object",
                                    "allOf": [{
                                          "$ref": "#/properties/ifcPerson"
                                    }]
                              },
                              "theOrganization": {
                                    "type": "object",
                                    "allOf": [{
                                          "$ref": "#/properties/ifcOrganization"
                                    }]
                              },
                              "roles": {
                                    "type": "null"
                              }
                        },
```

```
                        "required":   ["ifc",   "thePerson",   "theOrganization",
"roles"]
                },
                "ifcOrganization": {
                        "type": "object",
                        "properties": {
                                "instanceId": {
                                        "type": "integer"
                                },
                                "ifc": {
                                        "type": "string"
                                },
                                "identification": {
                                        "oneOf": [{
                                                        "type": "null"
                                                },
                                                {
                                                        "type": "string",
                                                        "maxLength": 255
                                                }
                                        ]
                                },
                                "name": {
                                        "type": "string",
                                        "maxLength": 255
                                },
                                "description": {
                                        "type": ["string", "null"]
                                },
                                "roles": {
                                        "type": "null"
                                },
                                "addresses": {
                                        "type": "null"
                                }
                        },
                        "required":     ["ifc",     "identification",     "name",
"description", "roles", "addresses"]
                },
                "ifcApplication": {
                        "type": "object",
                        "properties": {
                                "instanceId": {
                                        "type": "integer"
                                },
                                "ifc": {
                                        "type": "string"
                                },
                                "applicationDeveloper": {
                                        "type": "object",
                                        "allOf": [{
                                                "$ref": "#/properties/ifcOrganization"
                                        }]
                                },
                                "version": {
                                        "type": "string",
                                        "maxLength": 255
                                },
                                "applicationFullName": {
                                        "type": "string",
                                        "maxLength": 255
                                },
                                "applicationIdentifier": {
```

```
                                    "type": "string",
                                    "maxLength": 255
                             }
                      },
                      "required":  ["ifc",  "applicationDeveloper",  "version",
"applicationFullName", "applicationIdentifier"]
               },
               "IfcTelecomAddress": {
                      "type": "object",
                      "properties": {
                             "instanceId": {
                                    "type": "integer"
                             },
                             "ifc": {
                                    "type": "string"
                             },
                             "purpose": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "string",
                                                  "enum":   ["OFFICE",   "SITE",
"HOME", "DISTRIBUTIONPOINT", "USERDEFINED"]
                                           }
                                    ]
                             },
                             "description": {
                                    "type": ["string", "null"]
                             },
                             "userDefinedPurpose": {
                                    "type": ["string", "null"]
                             },
                             "telephoneNumbers": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "array",
                                                  "items": {
                                                         "type": "string",
                                                         "maxLength": 255
                                                  },
                                                  "minItems": 1
                                           }
                                    ]
                             },
                             "facsimileNumbers": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "array",
                                                  "items": {
                                                         "type": "string",
                                                         "maxLength": 255
                                                  },
                                                  "minItems": 1
                                           }
                                    ]
                             },
                             "pagerNumber": {
                                    "type": ["string", "null"]
```

```
                        },
                        "electronicMailAddresses": {
                              "oneOf": [{
                                          "type": "null"
                                    },
                                    {
                                          "type": "array",
                                          "items": {
                                                "type": "string",
                                                "maxLength": 255
                                          },
                                          "minItems": 1
                                    }
                              ]

                        },
                        "wwwHomePageURL": {
                              "type": ["string", "null"]
                        },
                        "messagingIDs": {
                              "oneOf": [{
                                          "type": "null"
                                    },
                                    {
                                          "type": "array",
                                          "items": {
                                                "type": "string",
                                                "maxLength": 255
                                          },
                                          "minItems": 1
                                    }
                              ]

                        }
                  },
                  "required":     ["ifc",     "purpose",     "description",
"userDefinedPurpose",  "telephoneNumbers",  "facsimileNumbers",  "pagerNumber",
"electronicMailAddresses", "wwwHomePageURL", "messagingIDs"]
            },
            "IfcActorRole": {
                  "type": "object",
                  "properties": {
                        "instanceId": {
                              "type": "integer"
                        },
                        "ifc": {
                              "type": "string"
                        },
                        "role": {
                              "type": "string",
                              "enum":     ["SUPPLIER",     "MANUFACTURER",
"CONTRACTOR",     "SUBCONTRACTOR",     "ARCHITECT",     "STRUCTURALENGINEER",
"COSTENGINEER",      "CLIENT",      "BUILDINGOWNER",      "BUILDINGOPERATOR",
"MECHANICALENGINEER",          "ELECTRICALENGINEER",          "PROJECTMANAGER",
"FACILITIESMANAGER",  "CIVILENGINEER",  "COMMISSIONINGENGINEER",  "ENGINEER",
"OWNER",  "CONSULTANT",  "CONSTRUCTIONMANAGER",  "FIELDCONSTRUCTIONMANAGER",
"RESELLER", "USERDEFINED"]
                        },
                        "userDefinedRole": {
                              "type": ["string", "null"]
                        },
                        "description": {
                              "type": ["string", "null"]
```

```
                        }
                    },
                    "required":     ["ifc",     "role",     "userDefinedRole",
"description"]
             },
             "ifcOwnerHistory": {
                    "type": "object",
                    "properties": {
                           "instanceId": {
                                  "type": "integer"
                           },
                           "ifc": {
                                  "type": "string"
                           },
                           "owningUser": {
                                  "type": "object",
                                  "allOf": [{
                                         "$ref":
"#/properties/ifcPersonAndOrganization"
                                  }]
                           },
                           "owningApplication": {
                                  "type": "object",
                                  "allOf": [{
                                         "$ref": "#/properties/ifcApplication"
                                  }]
                           },
                           "state": {
                                  "oneOf": [{
                                                "type": "null"
                                         },
                                         {
                                                "type": "string",
                                                "enum":           ["READWRITE",
"READONLY", "LOCKED", "READWRITELOCKED", "READONLYLOCKED"]
                                         }
                                  ]
                           },
                           "changeAction": {
                                  "oneOf": [{
                                                "type": "null"
                                         },
                                         {
                                                "type": "string",
                                                "enum":            ["NOCHANGE",
"MODIFIED", "ADDED", "DELETED", "NOTDEFINED"]
                                         }
                                  ]
                           },
                           "lastModifiedDate": {
                                  "oneOf": [{
                                                "type": "null"
                                         },
                                         {
                                                "type": "integer"
                                         }
                                  ]
                           },
                           "lastModifyingUser": {
                                  "oneOf": [{
                                                "type": "null"
                                         },
                                         {
```

```
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcPersonAndOrganization"
                                                }]
                                          }
                                    ]
                              },
                              "lastModifyingApplication": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcApplication"
                                                }]
                                          }
                                    ]
                              },
                              "creationDate": {
                                    "type": "integer"
                              }
                        },
                        "required":  ["ifc",  "owningUser",  "owningApplication",
"state",     "changeAction",     "lastModifiedDate",     "lastModifyingUser",
"lastModifyingApplication", "creationDate"]
                  },
                  "ifcObjectDefinition": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "globalId": {
                                    "type": "string",
                                    "maxLength": 22
                              },
                              "ownerHistory": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                          }
                                    ]
                              },
                              "name": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
```

```
                                    ]
                             },
                             "description": {
                                    "type": ["string", "null"]
                             }
                      },
                      "required": ["ifc", "globalId", "ownerHistory", "name",
"description"]
               },
               "ifcProduct": {
                      "type": "object",
                      "properties": {
                             "instanceId": {
                                    "type": "integer"
                             },
                             "ifc": {
                                    "type": "string"
                             },
                             "globalId": {
                                    "type": "string",
                                    "maxLength": 22
                             },
                             "ownerHistory": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "object",
                                                  "allOf": [{
                                                         "$ref":
"#/properties/ifcOwnerHistory"
                                                  }]
                                           }
                                    ]
                             },
                             "name": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "string",
                                                  "maxLength": 255
                                           }
                                    ]
                             },
                             "description": {
                                    "type": ["string", "null"]
                             },
                             "objectType": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "string",
                                                  "maxLength": 255
                                           }
                                    ]
                             },
                             "objectPlacement": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
```

```
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcLocalPlacement"
                                                }]
                                          }
                                    ]
                              },
                              "representation": {
                                    "type": "object",
                                    "allOf": [{
                                          "$ref":
"#/properties/ifcProductDefinitionShape"
                                    }]

                              }
                        },
                        "required": ["ifc", "globalId", "ownerHistory", "name",
"description", "objectType", "objectPlacement", "representation"]
                  },
                  "ifcElement": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "globalId": {
                                    "type": "string",
                                    "maxLength": 22
                              },
                              "ownerHistory": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                          }
                                    ]
                              },
                              "name": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
                                    ]
                              },
                              "description": {
                                    "type": ["string", "null"]
                              },
                              "objectType": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
```

```
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "objectPlacement": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcLocalPlacement"
                                                }]
                                        }
                                ]
                        },
                        "representation": {
                                "type": "object",
                                "allOf": [{
                                        "$ref":
"#/properties/ifcProductDefinitionShape"
                                }]

                        },
                        "tag": {
                                "type": "string",
                                "maxLength": 255
                        }
                },
                "required": ["ifc", "globalId", "ownerHistory", "name",
"description", "objectType", "objectPlacement", "representation", "tag"]
        },
        "ifcBuildingElement": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string",
                                "enum":       ["IFCCOLUMN",       "IFCBEAM",
"IFCFOOTING", "IFCSLAB", "IFCWALL"]
                        },
                        "globalId": {
                                "type": "string",
                                "maxLength": 22
                        },
                        "ownerHistory": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                        }
                                ]
                        },
```

```
                        "name": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
                                        "type": "string",
                                        "maxLength": 255
                                }
                                ]
                        },
                        "description": {
                                "type": ["string", "null"]
                        },
                        "objectType": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
                                        "type": "string",
                                        "maxLength": 255
                                }
                                ]
                        },
                        "objectPlacement": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
                                        "type": "object",
                                        "allOf": [{
                                                "$ref":
"#/properties/ifcLocalPlacement"
                                        }]
                                }
                                ]
                        },
                        "representation": {
                                "type": "object",
                                "allOf": [{
                                        "$ref":
"#/properties/ifcProductDefinitionShape"
                                }]

                        },
                        "tag": {
                                "type": "string",
                                "maxLength": 255
                        },
                        "predefinedType": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
                                        "type": "string",
                                        "enum":  ["COLUMN",  "PILASTER",
"USERDEFINED", "NOTDEFINED"]
                                },
                                {
                                        "type": "string",
                                        "enum":    ["BEAM",    "JOIST",
"HOLLOWCORE", "LINTEL", "SPANDREL", "T_BEAM"]
                                },
                                {
```

```
                                                "type": "string",
                                                "enum":  ["CAISSON_FOUNDATION",
"FOOTING_BEAM", "PAD_FOOTING", "PILE_CAP", "STRIP_FOOTING"]
                                          },
                                          {
                                                "type": "string",
                                                "enum":    ["FLOOR",    "ROOF",
"LANDING", "BASESLAB"]
                                          },
                                          {
                                                "type": "string",
                                                "enum":  ["MOVABLE",  "PARAPET",
"PARTITIONING", "PLUMBINGWALL", "SHEAR", "SOLIDWALL", "STANDARD", "POLYGONAL",
"ELEMENTEDWALL"]
                                          }
                                    ]
                              }
                        },
                        "required":  ["ifc",  "globalId",  "ownerHistory",  "name",
"description",   "objectType",   "objectPlacement",   "representation",   "tag",
"predefinedType"]
                  },
                  "ifcProjectionElement": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "globalId": {
                                    "type": "string",
                                    "maxLength": 22
                              },
                              "ownerHistory": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                          }
                                    ]
                              },
                              "name": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
                                    ]
                              },
                              "description": {
                                    "type": ["string", "null"]
                              },
                              "objectType": {
                                    "oneOf": [{
```

```
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "objectPlacement": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcLocalPlacement"
                                                }]
                                        }
                                ]
                        },
                        "representation": {
                                "type": "object",
                                "allOf": [{
                                        "$ref":
"#/properties/ifcProductDefinitionShape"
                                }]

                        },
                        "tag": {
                                "type": "string",
                                "maxLength": 255
                        },
                        "predefinedType": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "enum":         ["USERDEFINED",
"NOTDEFINED"]
                                        }
                                ]
                        }
                },
                "required":  ["ifc",  "globalId",  "ownerHistory",  "name",
"description",   "objectType",   "objectPlacement",   "representation",   "tag",
"predefinedType"]
        },
        "ifcElementAssembly": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "globalId": {
                                "type": "string",
                                "maxLength": 22
                        },
                        "ownerHistory": {
```

```
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                          }
                                    ]
                              },
                              "name": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
                                    ]
                              },
                              "description": {
                                    "type": ["string", "null"]
                              },
                              "objectType": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
                                    ]
                              },
                              "objectPlacement": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcLocalPlacement"
                                                }]
                                          }
                                    ]
                              },
                              "representation": {
                                    "type": "object",
                                    "allOf": [{
                                          "$ref":
"#/properties/ifcProductDefinitionShape"
                                    }]

                              },
                              "tag": {
                                    "type": "string",
                                    "maxLength": 255
                              },
                              "assemblyPlace": {
                                    "oneOf": [{
                                                "type": "null"
```

```
                                        },
                                        {
                                                "type": "string",
                                                "enum":   ["SITE",   "FACTORY",
"NOTDEFINED"]
                                        }
                                ]
                        },
                        "predefinedType": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "enum":  ["ACCESSORY_ASSEMBLY",
"ARCH",   "BEAM_GRID",   "BRACED_FRAME",   "GIRDER",   "REINFORCEMENT_UNIT",
"RIGID_FRAME", "SLAB_FIELD", "TRUSS", "USERDEFINED", "NOTDEFINED"]
                                        }
                                ]
                        }
                },
                "required": ["ifc", "globalId", "ownerHistory", "name",
"description",  "objectType",  "objectPlacement",  "representation",  "tag",
"assemblyPlace", "predefinedType"]
        },
        "ifcSpatialElement": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "globalId": {
                                "type": "string",
                                "maxLength": 22
                        },
                        "ownerHistory": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                        }
                                ]
                        },
                        "name": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "description": {
                                "type": ["string", "null"]
```

```
                        },
                        "objectType": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "objectPlacement": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcLocalPlacement"
                                                }]
                                        }
                                ]
                        },
                        "representation": {
                                "type": "object",
                                "allOf": [{
                                        "$ref":
"#/properties/ifcProductDefinitionShape"
                                }]

                        },
                        "longName": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        }
                },
                "required": ["ifc", "globalId", "ownerHistory", "name",
"description", "objectType", "objectPlacement", "representation", "longName"]
            },
            "ifcSite": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "globalId": {
                                "type": "string",
                                "maxLength": 22
                        },
                        "ownerHistory": {
                                "oneOf": [{
                                                "type": "null"
                                        },
```

```
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                        }
                                ]
                        },
                        "name": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "description": {
                                "type": ["string", "null"]
                        },
                        "objectType": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "objectPlacement": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcLocalPlacement"
                                                }]
                                        }
                                ]
                        },
                        "representation": {
                                "type": "object",
                                "allOf": [{
                                        "$ref":
"#/properties/ifcProductDefinitionShape"
                                }]

                        },
                        "longName": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
```

```
                        "compositionType": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
                                        "type": "string",
                                        "enum":  ["COMPLEX",  "ELEMENT",
"PARTIAL"]
                                }
                                ]
                        },
                        "refLatitude": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
                                        "type": "array",
                                        "items": {
                                                "type": "integer"
                                        },
                                        "minItems": 3,
                                        "maxItems": 4
                                }
                                ]
                        },
                        "refLongitude": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
                                        "type": "array",
                                        "items": {
                                                "type": "integer"
                                        },
                                        "minItems": 3,
                                        "maxItems": 4
                                }
                                ]
                        },
                        "refElevation": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
                                        "type": "number"
                                }
                                ]
                        },
                        "landTitleNumber": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
                                        "type": "string",
                                        "maxLength": 255
                                }
                                ]
                        },
                        "siteAddress": {
                                "oneOf": [{
                                        "type": "null"
                                },
                                {
```

```
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcPostalAddress"
                                                }]
                                          }
                                    ]
                              }
                        },
                        "required": ["ifc", "globalId", "ownerHistory", "name",
"description", "objectType", "objectPlacement", "representation", "longName",
"compositionType",     "refLatitude",     "refLongitude",     "refElevation",
"landTitleNumber", "siteAddress"]
                  },
                  "ifcBuilding": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "globalId": {
                                    "type": "string",
                                    "maxLength": 22
                              },
                              "ownerHistory": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "object",
                                                "allOf": [{
                                                      "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                          }
                                    ]
                              },
                              "name": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
                                    ]
                              },
                              "description": {
                                    "type": ["string", "null"]
                              },
                              "objectType": {
                                    "oneOf": [{
                                                "type": "null"
                                          },
                                          {
                                                "type": "string",
                                                "maxLength": 255
                                          }
                                    ]
                              },
```

```
                        "objectPlacement": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcLocalPlacement"
                                                }]
                                        }
                                ]
                        },
                        "representation": {
                                "type": "object",
                                "allOf": [{
                                        "$ref":
"#/properties/ifcProductDefinitionShape"
                                }]
                        },
                        "longName": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "compositionType": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "enum":  ["COMPLEX",  "ELEMENT",
"PARTIAL"]
                                        }
                                ]
                        },
                        "elevationOfRefHeight": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "number"
                                        }
                                ]
                        },
                        "elevationOfTerrain": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "number"
                                        }
                                ]
                        },
                        "buildingAddress": {
                                "oneOf": [{
                                                "type": "null"
                                        },
```

```
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcPostalAddress"
                                                }]
                                        }
                                ]
                        }
                },
                "required": ["ifc", "globalId", "ownerHistory", "name",
"description", "objectType", "objectPlacement", "representation", "longName",
"compositionType",        "elevationOfRefHeight",        "elevationOfTerrain",
"buildingAddress"]
        },
        "ifcBuildingStorey": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "globalId": {
                                "type": "string",
                                "maxLength": 22
                        },
                        "ownerHistory": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                        }
                                ]
                        },
                        "name": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "description": {
                                "type": ["string", "null"]
                        },
                        "objectType": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
```

```
                        },
                        "objectPlacement": {
                              "oneOf": [{
                                          "type": "null"
                                    },
                                    {
                                          "type": "object",
                                          "allOf": [{
                                                "$ref":
"#/properties/ifcLocalPlacement"
                                          }]
                                    }
                              ]
                        },
                        "representation": {
                              "type": "object",
                              "allOf": [{
                                    "$ref":
"#/properties/ifcProductDefinitionShape"
                              }]
                        },
                        "longName": {
                              "oneOf": [{
                                          "type": "null"
                                    },
                                    {
                                          "type": "string",
                                          "maxLength": 255
                                    }
                              ]
                        },
                        "compositionType": {
                              "oneOf": [{
                                          "type": "null"
                                    },
                                    {
                                          "type": "string",
                                          "enum":  ["COMPLEX",  "ELEMENT",
"PARTIAL"]
                                    }
                              ]
                        },
                        "elevation": {
                              "oneOf": [{
                                          "type": "null"
                                    },
                                    {
                                          "type": "number"
                                    }
                              ]
                        }
                  },
                  "required":  ["ifc",  "globalId",  "ownerHistory",  "name",
"description", "objectType", "objectPlacement", "representation", "longName",
"compositionType", "elevation"]
            },
            "ifcSIUnit": {
                  "type": "object",
                  "properties": {
                        "instanceId": {
                              "type": "integer"
                        },
                        "ifc": {
```

```
                                    "type": "string"
                             },
                             "unitType": {
                                    "type": "string",
                                    "enum":                    ["ABSORBEDDOSEUNIT",
"AMOUNTOFSUBSTANCEUNIT",            "AREAUNIT",           "DOSEEQUIVALENTUNIT",
"ELECTRICCAPACITANCEUNIT",   "ELECTRICCHARGEUNIT",   "ELECTRICCONDUCTANCEUNIT",
"ELECTRICCURRENTUNIT",     "ELECTRICRESISTANCEUNIT",     "ELECTRICVOLTAGEUNIT",
"ENERGYUNIT",       "FORCEUNIT",       "FREQUENCYUNIT",       "ILLUMINANCEUNIT",
"INDUCTANCEUNIT",  "LENGTHUNIT",  "LUMINOUSFLUXUNIT",  "LUMINOUSINTENSITYUNIT",
"MAGNETICFLUXDENSITYUNIT",  "MAGNETICFLUXUNIT",  "MASSUNIT",  "PLANEANGLEUNIT",
"POWERUNIT",     "PRESSUREUNIT",     "RADIOACTIVITYUNIT",     "SOLIDANGLEUNIT",
"THERMODYNAMICTEMPERATUREUNIT", "TIMEUNIT", "VOLUMEUNIT", "USERDEFINED"]
                             },
                             "prefix": {
                                    "oneOf": [{
                                                  "type": "null"
                                           },
                                           {
                                                  "type": "string",
                                                  "enum": ["EXA", "PETA", "TERA",
"GIGA",  "MEGA",  "KILO",  "HECTO",  "DECA",  "DECI",  "CENTI",  "MILLI",  "MICRO",
"NANO", "PICO", "FEMTO", "ATTO"]
                                           }
                                    ]
                             },
                             "name": {
                                    "type": "string",
                                    "enum":  ["AMPERE",  "BECQUEREL",  "CANDELA",
"COULOMB", "CUBIC_METRE", "DEGREE_CELSIUS", "FARAD", "GRAM", "GRAY", "HENRY",
"HERTZ", "JOULE", "KELVIN", "LUMEN", "LUX", "METRE", "MOLE", "NEWTON", "OHM",
"PASCAL",   "RADIAN",   "SECOND",   "SIEMENS",   "SIEVERT",   "SQUARE_METRE",
"STERADIAN", "TESLA", "VOLT", "WATT", "WEBER"]
                             }
                      },
                      "required": ["ifc", "unitType", "prefix", "name"]
               },
               "ifcUnitAssignment": {
                      "type": "object",
                      "properties": {
                             "instanceId": {
                                    "type": "integer"
                             },
                             "ifc": {
                                    "type": "string"
                             },
                             "units": {
                                    "type": "array",
                                    "items": {
                                           "type": "object",
                                           "allOf": [{
                                                  "$ref": "#/definitions/ifcUnit"
                                           }]
                                    },
                                    "minItems": 1
                             }
                      },
                      "required": ["ifc", "units"]
               },
               "ifcDerivedUnit": {
                      "type": "object",
                      "properties": {
                             "instanceId": {
```

```
                                    "type": "integer"
                             },
                             "ifc": {
                                    "type": "string"
                             },
                             "elements": {
                                    "type": "array",
                                    "items": {
                                           "type": "object",
                                           "allOf": [{
                                                  "$ref":
"#/properties/ifcDerivedUnitElement"
                                           }]
                                    },
                                    "minItems": 1
                             },
                             "unitType": {
                                    "type": "string",
                                    "enum":              ["ANGULARVELOCITYUNIT",
"AREADENSITYUNIT",      "COMPOUNDPLANEANGLEUNIT",      "DYNAMICVISCOSITYUNIT",
"HEATFLUXDENSITYUNIT",                                 "INTEGERCOUNTRATEUNIT",
"ISOTHERMALMOISTURECAPACITYUNIT",                     "KINEMATICVISCOSITYUNIT",
"LINEARVELOCITYUNIT",          "MASSDENSITYUNIT",          "MASSFLOWRATEUNIT",
"MOISTUREDIFFUSIVITYUNIT",  "MOLECULARWEIGHTUNIT",  "SPECIFICHEATCAPACITYUNIT",
"THERMALADMITTANCEUNIT",   "THERMALCONDUCTANCEUNIT",   "THERMALRESISTANCEUNIT",
"THERMALTRANSMITTANCEUNIT", "VAPORPERMEABILITYUNIT", "VOLUMETRICFLOWRATEUNIT",
"ROTATIONALFREQUENCYUNIT",         "TORQUEUNIT",         "MOMENTOFINERTIAUNIT",
"LINEARMOMENTUNIT",           "LINEARFORCEUNIT",           "PLANARFORCEUNIT",
"MODULUSOFELASTICITYUNIT",      "SHEARMODULUSUNIT",      "LINEARSTIFFNESSUNIT",
"ROTATIONALSTIFFNESSUNIT",                      "MODULUSOFSUBGRADEREACTIONUNIT",
"ACCELERATIONUNIT",            "CURVATUREUNIT",            "HEATINGVALUEUNIT",
"IONCONCENTRATIONUNIT",                    "LUMINOUSINTENSITYDISTRIBUTIONUNIT",
"MASSPERLENGTHUNIT",                     "MODULUSOFLINEARSUBGRADEREACTIONUNIT",
"MODULUSOFROTATIONALSUBGRADEREACTIONUNIT",   "PHUNIT",   "ROTATIONALMASSUNIT",
"SECTIONAREAINTEGRALUNIT",     "SECTIONMODULUSUNIT",     "SOUNDPOWERLEVELUNIT",
"SOUNDPOWERUNIT",        "SOUNDPRESSURELEVELUNIT",        "SOUNDPRESSUREUNIT",
"TEMPERATUREGRADIENTUNIT",                        "TEMPERATURERATEOFCHANGEUNIT",
"THERMALEXPANSIONCOEFFICIENTUNIT", "WARPINGCONSTANTUNIT", "WARPINGMOMENTUNIT",
"USERDEFINED"]
                             },
                             "userDefinedType": {
                                    "type": ["string", "null"]
                             }
                      },
                      "required":      ["ifc",      "elements",      "unitType",
"userDefinedType"]
               },
               "ifcDerivedUnitElement": {
                      "type": "object",
                      "properties": {
                             "instanceId": {
                                    "type": "integer"
                             },
                             "ifc": {
                                    "type": "string"
                             },
                             "unit": {
                                    "type": "object",
                                    "allOf": [{
                                           "$ref": "#/properties/ifcNamedUnit"
                                    }]
                             },
                             "exponent": {
```

```
                                    "type": "integer"
                              }
                        },
                        "required": ["ifc", "unit", "exponent"]
                  },
                  "ifcNamedUnit": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "dimensions": {
                                    "type": "object",
                                    "allOf": [{
                                          "$ref":
"#/properties/ifcDimensionalExponents"
                                    }]
                              },
                              "unitType": {
                                    "type": "string",
                                    "enum":                  ["ABSORBEDDOSEUNIT",
"AMOUNTOFSUBSTANCEUNIT",           "AREAUNIT",           "DOSEEQUIVALENTUNIT",
"ELECTRICCAPACITANCEUNIT",   "ELECTRICCHARGEUNIT",   "ELECTRICCONDUCTANCEUNIT",
"ELECTRICCURRENTUNIT",     "ELECTRICRESISTANCEUNIT",     "ELECTRICVOLTAGEUNIT",
"ENERGYUNIT",       "FORCEUNIT",       "FREQUENCYUNIT",       "ILLUMINANCEUNIT",
"INDUCTANCEUNIT",  "LENGTHUNIT",  "LUMINOUSFLUXUNIT",  "LUMINOUSINTENSITYUNIT",
"MAGNETICFLUXDENSITYUNIT",  "MAGNETICFLUXUNIT",  "MASSUNIT",  "PLANEANGLEUNIT",
"POWERUNIT",     "PRESSUREUNIT",     "RADIOACTIVITYUNIT",     "SOLIDANGLEUNIT",
"THERMODYNAMICTEMPERATUREUNIT", "TIMEUNIT", "VOLUMEUNIT", "USERDEFINED"]
                              }
                        },
                        "required": ["ifc", "dimensions", "unitType"]
                  },
                  "ifcDimensionalExponents": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "lengthExponent": {
                                    "type": "integer"
                              },
                              "massExponent": {
                                    "type": "integer"
                              },
                              "timeExponent": {
                                    "type": "integer"
                              },
                              "electricCurrentExponent": {
                                    "type": "integer"
                              },
                              "thermodynamicTemperatureExponent": {
                                    "type": "integer"
                              },
                              "amountOfSubstanceExponent": {
                                    "type": "integer"
                              },
```

```
                        "luminousIntensityExponent": {
                                "type": "integer"
                        }
                },
                "required":  ["ifc",  "lengthExponent",  "massExponent",
"timeExponent", "electricCurrentExponent", "thermodynamicTemperatureExponent",
"amountOfSubstanceExponent", "luminousIntensityExponent"]
        },
        "ifcMonetaryUnit": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "currency": {
                                "type": "string",
                                "maxLength": 255
                        }
                },
                "required": ["ifc", "currency"]
        },
        "ifcRelAggregates": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "globalId": {
                                "type": "string",
                                "maxLength": 22
                        },
                        "ownerHistory": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                        }
                                ]
                        },
                        "name": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "description": {
                                "type": ["string", "null"]
                        },
```

```
                        "relatingObject": {
                                "type": "object",
                                "allOf": [{
                                        "$ref":
"#/properties/ifcObjectDefinition"
                                }]
                        },
                        "relatedObjects": {
                                "type": "array",
                                "items": {
                                        "type": "object",
                                        "allOf": [{
                                                "$ref":
"#/properties/ifcObjectDefinition"
                                        }]
                                },
                                "minItems": 1
                        }
                },
                "required": ["ifc", "globalId", "ownerHistory", "name",
"description", "relatingObject", "relatedObjects"]
        },
        "ifcRelContainedInSpatialStructure": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "globalId": {
                                "type": "string",
                                "maxLength": 22
                        },
                        "ownerHistory": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                        }
                                ]
                        },
                        "name": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "description": {
                                "type": ["string", "null"]
                        },
                        "relatedElements": {
                                "type": "array",
```

```
                                    "items": {
                                          "type": "object",
                                          "allOf": [{
                                                "$ref":
"#/properties/ifcProduct"
                                          }]
                                    },
                                    "minItems": 1
                              },
                              "relatingStructure": {
                                    "type": "object",
                                    "allOf": [{
                                          "$ref":
"#/properties/ifcSpatialElement"
                                    }]
                              }
                        },
                        "required": ["ifc", "globalId", "ownerHistory", "name",
"description", "relatedElements", "relatingStructure"]
                  },
                  "ifcGeometricRepresentationSubContext": {
                        "type": "object",
                        "properties": {
                              "instanceId": {
                                    "type": "integer"
                              },
                              "ifc": {
                                    "type": "string"
                              },
                              "contextIdentifier": {
                                    "oneOf": [{
                                          "type": "null"
                                          },
                                          {
                                          "type": "string",
                                          "maxLength": 255
                                          }
                                    ]
                              },
                              "contextType": {
                                    "oneOf": [{
                                          "type": "null"
                                          },
                                          {
                                          "type": "string",
                                          "maxLength": 255
                                          }
                                    ]
                              },
                              "parentContext": {
                                    "type": "object",
                                    "allOf": [{
                                          "$ref":
"#/properties/ifcGeometricRepresentationContext"
                                    }]
                              },
                              "targetScale": {
                                    "oneOf": [{
                                          "type": "null"
                                          },
                                          {
                                          "type": "number",
                                          "minimum": 0,
```

```
                                                "exclusiveMinimum": false
                                        }
                                ]
                        },
                        "targetView": {
                                "type": "string",
                                "enum":     ["GRAPH_VIEW",     "SKETCH_VIEW",
"MODEL_VIEW",     "PLAN_VIEW",      "REFLECTED_PLAN_VIEW",     "SECTION_VIEW",
"ELEVATION_VIEW", "USERDEFINED", "NOTDEFINED"]
                        },
                        "userDefinedTargetView": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        }
                },
                "required":  ["ifc",  "contextIdentifier",  "contextType",
"parentContext", "targetScale", "targetView", "userDefinedTargetView"]
        },
        "ifcProject": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "globalId": {
                                "type": "string",
                                "maxLength": 22
                        },
                        "ownerHistory": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "object",
                                                "allOf": [{
                                                        "$ref":
"#/properties/ifcOwnerHistory"
                                                }]
                                        }
                                ]
                        },
                        "name": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "description": {
                                "type": ["string", "null"]
                        }
```

```
                    },
                    "required": ["ifc", "globalId", "ownerHistory", "name",
"description"]
             },
             "ifcPostalAddress": {
                    "type": "object",
                    "properties": {
                           "instanceId": {
                                  "type": "integer"
                           },
                           "ifc": {
                                  "type": "string"
                           },
                           "purpose": {
                                  "oneOf": [{
                                                "type": "null"
                                         },
                                         {
                                                "type": "string",
                                                "enum":   ["OFFICE",   "SITE",
"HOME", "DISTRIBUTIONPOINT", "USERDEFINED"]
                                         }
                                  ]
                           },
                           "description": {
                                  "type": ["string", "null"]
                           },
                           "userDefinedPurpose": {
                                  "oneOf": [{
                                                "type": "null"
                                         },
                                         {
                                                "type": "string",
                                                "maxLength": 255
                                         }
                                  ]
                           },
                           "internalLocation": {
                                  "oneOf": [{
                                                "type": "null"
                                         },
                                         {
                                                "type": "string",
                                                "maxLength": 255
                                         }
                                  ]
                           },
                           "addressLines": {
                                  "oneOf": [{
                                                "type": "null"
                                         },
                                         {
                                                "type": "array",
                                                "items": {
                                                       "type": "string",
                                                       "maxLength": 255
                                                },
                                                "minItems": 1
                                         }
                                  ]
                           },
                           "postalBox": {
                                  "oneOf": [{
```

```
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "town": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "region": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "postalCode": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        },
                        "country": {
                                "oneOf": [{
                                                "type": "null"
                                        },
                                        {
                                                "type": "string",
                                                "maxLength": 255
                                        }
                                ]
                        }
                },
                "required":     ["ifc",     "purpose",     "description",
"userDefinedPurpose", "internalLocation", "addressLines", "postalBox", "town",
"region", "postalCode", "country"]
        },
        "ifcRelProjectsElement": {
                "type": "object",
                "properties": {
                        "instanceId": {
                                "type": "integer"
                        },
                        "ifc": {
                                "type": "string"
                        },
                        "globalId": {
```

```
                                    "type": "string",
                                    "maxLength": 22
                             },
                             "ownerHistory": {
                                    "oneOf": [{
                                                 "type": "null"
                                           },
                                           {
                                                 "type": "object",
                                                 "allOf": [{
                                                       "$ref":
"#/properties/ifcOwnerHistory"
                                                 }]
                                           }
                                    ]
                             },
                             "name": {
                                    "oneOf": [{
                                                 "type": "null"
                                           },
                                           {
                                                 "type": "string",
                                                 "maxLength": 255
                                           }
                                    ]
                             },
                             "description": {
                                    "type": ["string", "null"]
                             },
                             "relatingElement": {
                                    "type": "object",
                                    "allOf": [{
                                           "$ref": "#/properties/ifcElement"
                                    }]
                             },
                             "relatedFeatureElement": {
                                    "type": "object",
                                    "allOf": [{
                                           "$ref":
"#/properties/ifcProjectionElement"
                                    }]
                             }
                      },
                      "required": ["ifc", "globalId", "ownerHistory", "name",
"description", "relatingElement", "relatedFeatureElement"]
               }
       }
}
```

REFERENCES

[1] NIST, "US Government Cloud Computing Technology Roadmap, Volume I and II," National Institute of Standards and Technology, 2014.

[2] Y. Jadeja and K. Modi , "Cloud Computing - Concepts, Architecture and Challenges," in *nternational Conference on Computing Electronics and Electrical Technologies*, Nagercoil, India, 2012.

[3] B. Rimal , E. Choi and I. Lumb , "A taxonomy and survey of cloud computing systems," in *The fifth international joint conference on INC, IMS and IDC*, 2009.

[4] D. Juan and Q. Zheng, "Cloud and Open BIM-Based Building Information Interoperability Research," *Journal of Service Science and Management,* vol. 7, pp. 47-56, 2014.

[5] J. Wong, X. Wang, H. Li, G. Chan and H. Li, "A Review of Cloud-Based BIM Technology in the Construction Sector," *Journal of Information Technology in Construction*, vol. 19, no. Special Issue: BIM Cloud-Based Technology in the AEC Sector: Present Status and Future Trends, pp. 281-291, 2014.

[6] J. Zhang, Q. Liu, F. Yu, Z. Hu and W. Zhao, "A Framework of Cloud-computing-based BIM Service for Building Lifecycle," in *International Conference on Computing in Civil and Building Engineering*, 2014.

[7] A. Redmond, A. Hore, M. Alshawi and R. West, "Exploring How Information Exchange Can Be Enhanced through Cloud BIM," *Automation in Construction*, vol. 24, pp. 175-183, 2012.

[8] A. Mahamadu, L. Mahdjoubi and C. Booth, "Challenges to BIM-cloud integration: Implication of security issues on secure collaboration," in *5th IEEE International Conference on Cloud Computing Technology and Science*, Bristol, UK, 2013.

[9] W. Wu and R. Issa, "Leveraging Cloud-BIM for LEED Automation," *Journal of Information Technology in Construction*, vol. 17, pp. 367-384, 2012.

[10] L. Berlo, "BIM Service interface exchange (BIMSie)," 2013. [Online]. Available: https://www.nibs.org/?page=bsa_bimsie. [Accessed May 2017].

[11] M. Shafiq , J. Matthews and S. Lockley , "A Study of BIM Collaboration Requirements and Available Features in Existing Model Collaboration Systems," *Journal of Information Technology in Construction*, vol. 18, pp. 148-161, 2013.

[12] J. Beetz, L. Berlo, R. Laat and P. Bonsma, "Advances in the development and application of an open source model server for building information," in *CIB W078 – Information Technology for Construction*, Sophia Antipolis, France, 2011.

[13] E. Curry, J. O'Donnell, E. Corry, S. Hasan, M. Keane and S. O'Riain, "Linking building data in the cloud: Integrating cross-domain building data using linked data," *Advanced Engineering Informatics*, vol. 27, no. 2, p. 206–219, 2013.

[14] J. Yang, R. Anand, S. Hobson, J. Lee, Y. Wang and J. Xu, "Data Service Portal for Application Integration in Cloud Computing," in *IEEE- 8th International Conference & Expo on Emerging Technologies for a Smarter World (CEWIT)*, New York, NY, 2011.

[15] Flux, "Flux Overview," 2017. [Online]. Available: https://community.flux.io/content/kbentry/1258/flux-overview.html. [Accessed May 2017].

[16] T. Dillon, C. Wu and E. Chang, "Cloud Computing: Issues and Challenges," in *IEEE International Conference on Advanced Information Networking and Applications*, Perth, Australia, 2010.

[17] D. Petcu , C. Craciun and M. Rak , "Towards a cross platform Cloud API—components for Cloud federation," in *CLOSER, 1st International Conference on Cloud Computing and Services Science*, Noordwijkerhout, Netherlands, 2011.

[18] K. Afsari, C. Eastman and D. Shelden, "Cloud-based BIM Data Transmission: Current Status and Challenges," in *33rd International Symposium on Automation and Robotics in Construction (ISARC)*, Auburn, Alabama, 2016.

[19] A. Almutairi, M. Sarfraz, S. Basalamah, W. Aref and G. Ghafoor, "A Distributed Access Control Architecture for Cloud Computing," *IEEE Software,* vol. 29, no. 2, pp. 36-44, 2012.

[20] C. Eastman, Y. Jeong, , R. Sacks and I. Kaner, "Exchange Model and Exchange Object Concepts for Implementation of National BIM Standards," *Journal of Computing in Civil Engineering,* vol. 24, no. 1, pp. 25-34, 2010.

[21] buildingSMART International, "IFC Overview summary," buildingSMART International, 2008. [Online]. Available: http://www.buildingsmart-tech.org/specifications/ifc-overview. [Accessed May 2017].

[22] ISO, "Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries," International Organization for Standardization (ISO), 2013.

[23] D. Schenck and P. Wilson , Information Modeling: The EXPRESS Way, Oxford University Press, 1994.

[24] C. Eastman, R. Sacks, I. Panushev , S. Aram and E. Yagmur , "The National Building Information Modeling Standards: Information Delivery Manual for Precast Concrete," The Charles Pankaw Foundation and PCI, 2009.

[25] I. Panushev, C. Eastman, R. Sacks, M. Venugopal and S. Aram, "Development of the National BIM Standard (NBIMS) For Precast/Prestressed Concrete," in *CIB W78 Workshop Proceedings*, Cairo, Egypt, 2010.

[26] C. Eastman , I. Panushev, R. Sacks, M. Venugopal, S. Aram, R. See and E. Yagmur, "A Guide for Development and Preparation of a National BIM Exchnage Standard," Charles Pankow Foundation, 2011.

[27] M. Venugopal, C. Eastman, R. Sacks and J. Teizer, "Improving the Robustness of Model Exchanges Using Product Modeling Concepts for IFC Schema," *Computing in Civil Engineering,* pp. 611-618, 2011.

[28] buildingSMART International, "An Integrated Process for Delivering IFC Based Data Exchange," buildingSMART International User Group, 2012.

[29] National Institute of Building Sciences, "National BIM Standard- United States. Version 3," National Institute of Building Sciences buildingSMART alliance, 2015.

[30] CogniFit, "Neuroplasticity Structure and organization," [Online]. Available: https://www.cognifit.com/brain-plasticity-and-cognition. [Accessed May 2017].

[31] R. Hall, "AP Psychology," 1867. [Online]. Available: http://www.rhsmpsychology.com/Handouts/synapse.htm. [Accessed June 2017].

[32] J. Beetz, L. Berlo, R. Laat and P. Halem, "BIMSERVER.ORG - An Open Source IFC Model Serer," in *CIB W78: 27th International Conference*, Cairo. Egypt, 2010.

[33] G. Zeiss, "BIMSie: a standard API for BIM web services in the cloud,," 19 March 2014. [Online]. Available: http://geospatial.blogs.com/geospatial/2014/03/bimsie-a-standard-api-for-bim-web-services-in-the-cloud.html.

[34] E. Curry, J. O'Donnell and E. Corry, "Building Optimisation using Scenario Modeling and Linked Data," in *First InternationalWorkshop on Linked Data in Architecture and Construction*, Ghent University, Ghent, Belgium, 2012.

[35] J. Beetz, J. Leeuwen and B. Vries, "IfcOWL: A case of transforming EXPRESS schemas," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 23, pp. 89-101, 2009.

[36] P. Pauwels and W. Terkaj, "EXPRESS to OWL for construction industry: Towards a recommendable," *Automation in Construction*, vol. 63, pp. 100-133, 2016.

[37] N. Loutas, E. Kamateri and K. Tarabanis, "A Semantic Interoperability Framework for Cloud Platform as a Service," in *Third IEEE International Conference on Coud Computing Technology and Science*, 2011.

[38] P. T. Endo, G. Gon¸calves , J. Kelner and D. Sadok , "A Survey on Open-source Cloud Computing Solutions," in *VIII Workshop em Clouds*, 2010.

[39] K. Fall and W. Stevens, "Chapter 1 Introduction," in *TCP/IP Illustrated- Volume 1*, Ann Arbor, Michigan, Pearson Education, 2012, pp. 2-28.

[40] C. Gong, J. Liu, Q. Zhang, H. Chen and Z. Gong, "The Characteristics of Cloud Computing," in *IEEE: 39th International Conference on Parallel Processing Workshops*, San Diego, CA, 2010.

[41] G. Lewis, "The Role of Standards in Cloud Computing Interoperability," in *IEEE: 46th International Conference on System Sciences*, Hawaii, 2013.

[42] N. Antonopoulos and L. Gillam, Cloud Computing: Principles, Systems and Applications, London: Springer-Verlag London, 2010.

[43] D. Bernstein , E. Ludvigson , K. Sankar , S. Diamond and M. Morrow , "Blueprint for the intercloud – protocols and formats for cloud computing interoperability," in *IEEE: 4th International Conference on Internet and Web Applications and Services (ICIW 2009)*, Venice/Mestre, Italy, 2009.

[44] N. Loutas, E. Kamateri and T. Konstantinos, "Cloud computing interoperability: the state of play," in *IEEE: Third International Conference on Cloud Computing Technology and Science (CloudCom)*, Athens, Greece, 2011.

[45] IBM, "TCP/IP Tutorial and Technical Overview," International Technical Support Organization, 2006.

[46] A. Sumaray and S. Makki, "A comparison of data serialization formats for optimal efficiency on a mobile platform," in *6th International Conf. on Ubiquitous Information Management and Communication*, Kuala Lumpur. Malaysia, 2012.

[47] G. Wang, "Improving Data Transmission in Web Applications via the Translation between XML and JSON," in *IEEE: Third International Conference on Communications and Mobile Computing*, Qingdao, China, 2011.

[48] K. Maeda, "Performance evaluation of object serialization libraries in XML, JSON and binary formats," in *IEEE: 2nd Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, Bangkok, Thailand, 2012.

[49] D. Petcu, "Consuming Resources and Services from Multiple Clouds- From Terminology to Cloudware Support," *Journal of Grid Computing*, vol. 12, p. 321–345, 2014.

[50] S. Dippl, M. L. Jaeger, A. Luhn, A. Shulman-Peleg and G. Vernik, "Towards Federation and Interoperability of Cloud Storage Systems," in *Data Intensive Storage Services for Cloud Environments*, Business Science Reference, 2013, p. 60.

[51] M. M. Mosbah, H. Soliman and M. Abou El-Nasr, "Current Services in Cloud Computing: A Survey," *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol.3,No.5,* 2013.

[52] K. Afsari, C. Eastman and D. Shelden, "Data Transmission Opportunities for Collaborative Cloud-Based Building Information Modeling," in *SIGraDi 2016, XX Congress of the Iberoamerican Society of Digital Graphics*, Buenos Aires, Argentina, 2016.

[53] P. Adamczyk, P. Smith, R. Johnson and M. Hafiz, "REST and Web Services: In Theory and in Practice," in *REST: From Research to Practice*, New York, NY, Springer, 2011, pp. 35-57.

[54] J. Wix, N. Nisbet and T. Liebich, "Using Constraints to Validate and Check Building Information Models," in *eWork and eBusiness in Architecture, Engineering and Construction*, Taylor & Francis, 2009, pp. 467-476.

[55] R. Vanlande, C. Nicolle and C. Cruz, "IFC and building lifecycle management," *Automation in Construction*, vol. 18, no. 1, pp. 70-78, 2008.

[56] buildingSMART, "IFC Technology," buildingSMART International, [Online]. Available: http://www.buildingsmart-tech.org/specifications/ifc-overview/ifc-technology . [Accessed October 2016].

[57] D. Schneck and P. Wilson, Information Modeling: The EXPRESS Way, Oxford University Press, 1994.

[58] N. Nisbet and L. Thomas, "ifcXML Implementation Guide," International Alliance for Interoperability Modeling Support Group, 2007.

[59] ISO, ISO 10303-21: Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure, ISO TC 184/SC4/WG11 N287, 2004.

[60] buildingSMART, "Industry Foundation Classes Release 4 (IFC4)," buildingSMART, 2013. [Online]. Available: http://www.buildingsmart-tech.org/ifc/IFC4/final/html/. [Accessed October 2016].

[61] N. Nurseitov, M. Paulson, R. Reynolds and C. Izurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study," in *Proceedings of the ISCA 22nd International Conference on Computer Applications in Industry and Engineering, CAINE*, San Francisco, California, 2009.

[62] buildingSMART, "ifcXML4 Release summary," buildingSMART international, [Online]. Available: http://www.buildingsmart-tech.org/specifications/ifcxml-releases/ifcxml4-release. [Accessed October 2016].

[63] D. Peng, L. Cao and W. Xu, "Using JSON for Data Exchanging in Web Service Applications," *Journal of Computational Information Systems* , vol. 7, no. 16, pp. 5883-5890, 2011.

[64] M. Gerhart, J. Bayer, J. M. Hofner and M. Boger, "Approach to Define Highly Scalable Metamodels Based on JSON," in *BigMDE 2015- 3rd Workshop on Scalable Model Driven*, L'Aquila, Italy, 2015.

[65] D. Vohra, Ajax in Oracle JDeveloper, Berlin Heidelberg: Springer, 2008.

[66] va3c, "Web Viewer for AEC Models," Open Source MIT License, [Online]. Available: https://va3c.github.io/. [Accessed October 2016].

[67] Autodesk, "Autodesk Forge Viewer," Autodesk, [Online]. Available: https://developer-autodesk.github.io/. [Accessed October 2016].

[68] L. V. Berlo, M. V. D. Jagt-Deutekom, R. V. Walsum, W. Klein and I. Mullers, "ELASSTIC project- Report on Improved Usage of BIM Technology," TNO, Delft, The Netherlands, 2016.

[69] GitHub, "BIMserver/Documentation/files/Test 1 Shell.json," [Online]. Available: https://github.com/opensourceBIM/BIMserver/blob/master/Documentation/files/Test%201%20Shell.json. [Accessed October 2016].

[70] GitHub, "BIMserver-JavaScript-API/js/ifc4.js," [Online]. Available: https://github.com/opensourceBIM/BIMserver-JavaScript-API/blob/master/js/ifc4.js. [Accessed October 2016].

[71] A. Wright, "JSON Schema: A Media Type for Describing JSON Documents-Internet Draft," Internet Engineering Task Force (IETF)-, 2016.

[72] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," Internet Engineering Task Force (IETF), 2014.

[73] The Open BIM Standards, "bimJSON," openbimstandards.org, [Online]. Available: http://openbimstandards.org/standards/bimjson/. [Accessed October 2016].

[74] IETF, "GEOJSON," Internet Engineering Task Force (IETF), [Online]. Available: http://geojson.org/. [Accessed October 2016].

[75] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen and T. Schaub, "The GeoJSON Format," Internet Engineering Task Force (IETF), August 2016. [Online]. Available: https://tools.ietf.org/html/rfc7946. [Accessed October 2016].

[76] Georgia Tech, "Precast Concrete BIM Standard," Georgia Tech, 2016. [Online]. Available: http://dcom.arch.gatech.edu/pci. [Accessed October 2016].

[77] K. Afsari and C. M. Eastman, "Consolidated Exchange Models for Implementing Precast Concrete Model View Definition," in *33rd International Symposium on Automation and Robotics in Construction (ISARC 2016)* , Auburn, Alabama, 2016.

[78] JSON Schema, "JSON Schema Specification," [Online]. Available: http://json-schema.org/documentation.html. [Accessed October 2016].

[79] M. Droettboom, Understanding JSON Schema, Space Telescope Science Institute, 2015.

[80] N. Maynard, "JSON Schema Lint," The MIT License (MIT), 2014. [Online]. Available: http://jsonschemalint.com/#/version/draft-05/markup/json. [Accessed October 2016].

[81] E. Poberezkin, "Ajv: Another JSON Schema Validator," The MIT License (MIT), 2015. [Online]. Available: https://github.com/epoberezkin/ajv. [Accessed October 2016].

[82] D. Petcu, C. Craciun, M. Neagul, I. Lazcanotegui and M. Rak, "Building an Interoperability API for Sky Computing," in *International Conference on High Performance Computing and Simulation (HPCS)*, Istanbul, 2011.

[83] Z. Zhang, C. Wu and D. W. Cheung, "A Survey on Cloud Interoperability-Taxonomies, Standards, and Practice," *ACM SIGMETRICS Performance Evaluation Review,* vol. 40, pp. 13-22, 2013.

[84] E. Wilde, "Putting Things to REST," UCB iSchool Report, Berkeley, 2007.

[85] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Journal of Future Generation Computer Systems*, vol. 29, pp. 1645-1660, 2013.

[86] D. Guinard, V. Trifa and E. Wilde, "A Resource Oriented Architecture for the Web of Things," in *Internet of Things (IOT)*, Tokyo, 2010.

[87] R. Fielding, *Architectural styles and the design of network-based software architectures*, University of California, Irvine, 2000.

[88] C. Pautasso and E. Wilde, "Introduction," in *REST: From Research to Practice*, New York, Springer, 2011, pp. 1-18.

[89] D. Guinard, M. Mueller and V. Trifa, "RESTifying Real-World Systems: A Practical Case Study in RFID," in *REST From Research to Practice*, New York, Springer, 2011, pp. 359-379.

[90] M. Hausenblas, "On Entities in theWeb of Data," in *REST: From Research to Practice*, Springer, 2011, pp. 425-440.

[91] C. Fan, J. Dobos, T. Scully, J. Daniel, G. Ducatel and R. Rowlingson, "3D Repo in a Secure Cloud Environment; a Case Study," 3drepo.org, London, UK, 2017.

[92] T. Scully, J. Dobos, T. Sturm and Y. Jung, "3drepo.io: Building the Next Generation Web3D Repository with AngularJS and X3DOM," ACM, London, UK, 2015.

[93] J. Dobos, K. Sons, D. Rubinstein, P. Slusallek and A. Steed, "XML3DRepo: A REST API for Version Controlled 3D Assets on the Web," 3drepo.org, London, UK, 2013.

[94] J. Dobos and A. Steed, "Revision Control Framework for 3D Assets," UCLA, London, UK, 2012.

[95] G. Jansen, "Resources, RESTful API design," 2011. [Online]. Available: http://restful-api-design.readthedocs.org/en/latest/resources.html.

[96] P. Subramaniam, "REST API Design - Resource Modeling," 2015. [Online]. Available: https://www.thoughtworks.com/insights/blog/rest-api-design-resource-modeling.

[97] buildingSMART, "Model View Definition Summary," [Online]. Available: http://www.buildingsmart-tech.org/specifications/ifc-view-definition. [Accessed April 2017].

[98] J. Hietanen , "IFC Model View Definition Format," International Alliance for Interoperability , 2006.

[99] D. Perry, A. Porter and L. Votta, "Empirical Studies of Software Engineering: A Roadmap," in *Conference on The Future of Software Engineering*, Limerick, Ireland, 2000.

[100] S. Pfleeger, "Experimental Design and Analysis in Software Engineering- Part 2: How to Set Up an Experiment," *Software Engineering Notes*, vol. 20, no. 1, pp. 22-26, 1995.

[101] D. Wallace, L. Ippolito and B. Cuthill, Reference Information for the Software Verification and Validation Process, NIST, 1996.

[102] M. Belsky , C. Eastman , R. Sacks, M. Venugopal , S. Aram and D. Yang, "Interoperability for precast concrete building models," *PCI Journal,* vol. 59, no. 2, p. 144, 2014.

[103] Z. Parker, S. Poe and S. Vrbsky , "Comparing NoSQL MongoDB to an SQL DB," in *51st ACM Southeast Conference*, Savannah, Georgia, 2013.

[104] S. Stonebraker, "Scalable SQL and NoSQL Data Stores," *Communications of the ACM,* vol. 53, no. 4, pp. 10-11, 2010.

[105] R. Cattell, "Scalable SQL and NoSQL Data Stores," *ACM SIGMOD Record,* vol. 39, no. 4, pp. 12-27, 2010.

[106] SmartBear, "What is SoapUI," 2015. [Online]. Available: http://www.soapui.org/about-soapui/what-is-soapui-.html.

[107] C. Severance , "Discovering JavaScript Object Notation," *Computer,* vol. 45, pp. 6-8, 2012.

[108] N. Serrano and J. P. Aroztegi, "Ajax Frameworks for Interactive Web Apps," *Software Technology, IEEE,* pp. 12 - 14, 2007.

[109] K. Afsari, C. Eastman and D. Castro, "JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange," *Automation in Construction,* vol. 77, pp. 24-51, 2016.